Graphics Systems and Models

Chapter 1

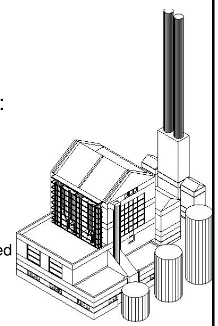
- Introduction:
  - Computer Graphics
    - What is it?
  - Overview of what we will cover:
    - A graphics overview
    - Graphics Theory
    - A graphics Software System: OpenGL
  - Our approach will be top-down.
    - We want you to start writing application programs that generate graphical output as quickly as possible

1. Applications of Computer Graphics

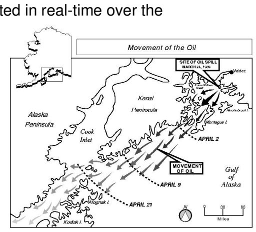
- The development of Computer Graphics has been driven by the needs of the user community and by the advances in hardware and software.
- Applications can be split into four major areas:
  - Display of information
  - Design
  - Simulation
  - User Interfaces

■ 1.1 Display of Information

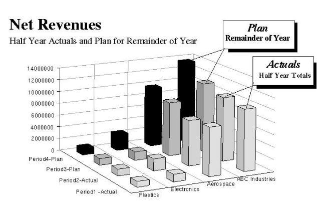
- Classical graphics techniques arose as a medium to convey information among people:
  - 4,000 years ago -- Babylonians: floor plans of buildings on stones
  - 2,000 years ago -- Greeks: Architectural ideas
    - – Now we have Computer-based drafting programs



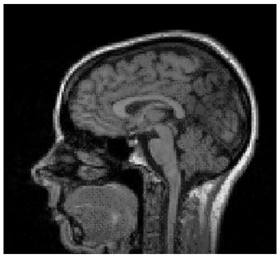
- For centuries -- Cartographers have developed maps to display celestial and geographical information.
  - – Now maps can be developed and manipulated in real-time over the internet.



- Over the past 100 years -- workers in statistics have explored techniques for generating plots to convey information
  - – Now we have computer plotting packages



- Medicine poses interesting and important data-analysis problems
  - CAT Scans, MRI's ultrasound, and other 3D data producing technologies



- The field of Scientific Visualization provides graphical tools that help these researchers and others interpret the vast quantities of data generated

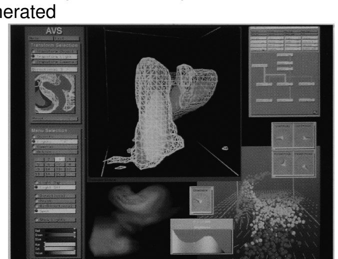


Plate 1. Severe tornadic storm, by R. Wilhelmson, L. Wicker, and C. Shaw, NCSA, University of Illinois. (Application Visualization System by Stardent Computer.)

■ 1.2 Design

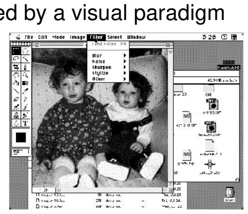
- Engineering and Architecture are concerned with design
  - starting with a set of specification
  - seek a cost-effective (and esthetic) solution
  - This is an iterative process
- The power of interacting with images on the screen
  - has been known for at least 40 years.
  - and today the use of interactive tools pervades the CAD field in areas such as architecture and VLSI design

■ 1.3 Simulation

- When did graphics begin to be used?
- Why?
- The field of VR has opened up many new horizons.
  - Same (or different) image in each eye
  - position tracking
  - interactive devices
  - This has led to training capabilities
    - NASA research grant

■ 1.4 User Interfaces

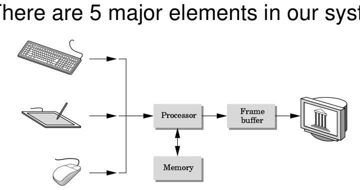
- Our interaction with computers has become dominated by a visual paradigm that includes:
  - windows,
  - icons,
  - menus, and
  - a pointing device



  - We have become so accustomed to this style of interface that we often forget that what we are doing is working with computer graphics.

2. A Graphics System

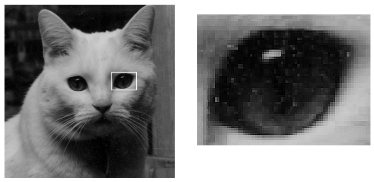
- There are 5 major elements in our system



  - Processor, Memory, Frame Buffer, Input Devices, Output Devices

■ 2.1 Pixels and the Frame Buffer

- At present, almost all graphics systems are raster-based
  - A picture is produced from an array (the raster) of picture elements (pixels)

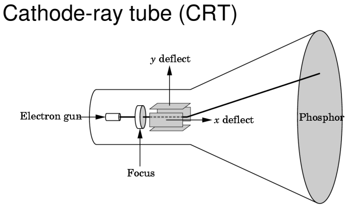


  - Def: depth of the frame buffer

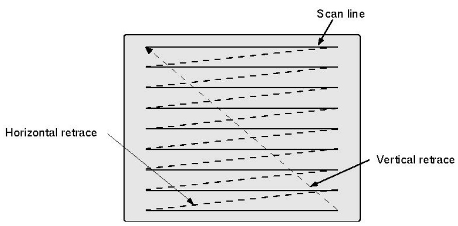
- Definitions:
  - Depth of the frame buffer
    - 1-bit
    - 8-bit deep
    - full-color is 24-bit or more
  - Resolution
    - the number of pixels in the frame buffer
  - Rasterization or scan conversion
    - The converting of geometric primitives into pixel assignments in the frame buffer

■ 2.2 Output Devices

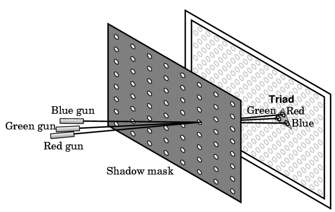
- The dominate type of display is the Cathode-ray tube (CRT)



- Def: refresh rate



- Color CRT's
  - have three different colored phosphors arranged in small groups (typically triads).

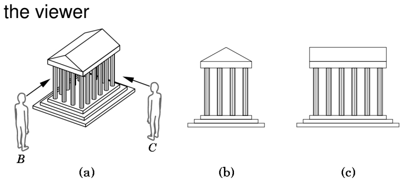


- 2.3 Input Devices
  - Most graphics systems provide a keyboard and at least one other input device
    - mouse
    - joystick
    - data tablet
  - We will study these devices in Chapter 3

3. Images: Physical and Synthetic

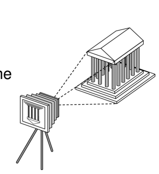
- The Usual pedagogical approach:
  - construct raster images
    - simple 2D entities (points, lines, polygons)
  - Define objects based upon 2D
  - Because such functionality is supported by most present computer graphics systems, we are going to learn to create images here, rather than expand a limited model.

- 3.1 Objects and Viewers
  - Two basic entities must be part of any image-formation process:
    - the object
    - the viewer



- The object exists in space independent of any image-formation process, and of any viewer.

- What we will study Now
  - Both the object and the viewer exist in a 3D world. However, the image they define is 2D
  - The process by which the specification of the object is combined with the specification of the viewer to produce an image is the essence of image formation.

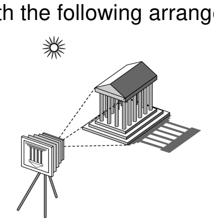


- Future:
  - In Chapter 2, we show how OpenGL allows us to build simple objects
  - In Chapter 8 we learn how to define objects in a manner that incorporates relationships among objects.

■ 3.2 Light and Images

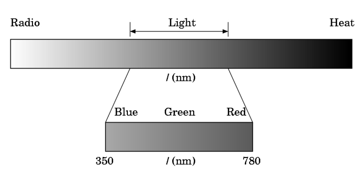
- Much information was missing from the preceding picture:
  - We have yet to mention light!
    - If there were no light sources the objects would be dark, and there would be nothing visible in our image.
  - We have not mentioned how color enters the picture
  - Or, what are the effects of different kinds of surfaces on the objects.

- Taking a more physical approach, we can start with the following arrangement:



  - The details of the interaction between light and the surfaces of the object determine how much light enters the camera.

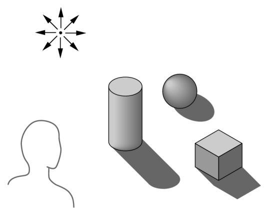
- Light is a form of electromagnetic radiation:



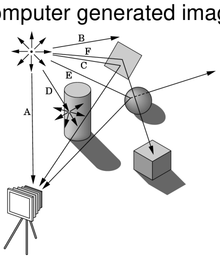
  - The details of the interaction between light and the surfaces of the object determine how much light enters the camera.

■ 3.3 Ray Tracing

- We can start building an imaging model by following light from a source.

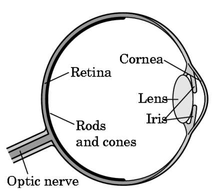


- Ray tracing is an image formation technique that is based on these ideas and that can form the basis for producing computer generated images.



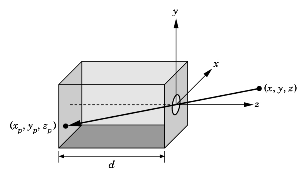
4. The Human Visual System

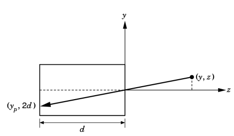
- Our extremely complex visual system has all the components of a physical imaging system, such as a camera or a microscope.

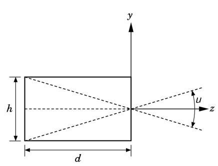


5. The Pinhole Camera

- A Pinhole camera is a box
  - with a small hole in the center on one side,
  - and the film on the opposite side



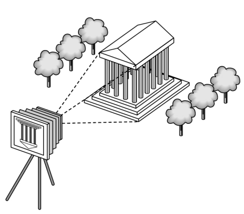




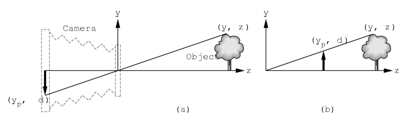
– Pinhole camera near Cliff House in San Francisco

6. The Synthetic Camera Model

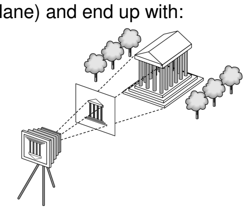
- Consider the imaging system shown here:



- A few Basic principles:
  - First, the specification of the objects is independent of the specification of the viewer.
    - In a graphics library we would expect separate functions for specifying objects and the viewer.
  - Second, we can compute the image using simple trigonometric calculations

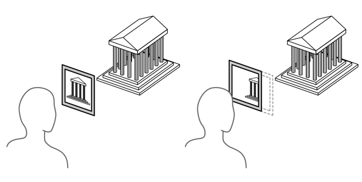


- Because of this, we can move the image plane in front of the lens (call this the projection plane) and end up with:



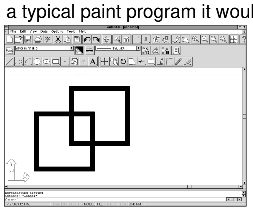
- In Chapter 5, we discuss this process in detail and derive the relevant mathematical formulas

- We must also consider the limited size of the image.
- Therefore, we must discuss clipping:



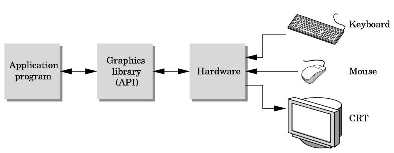
7. The Programmer's Interface

- There are numerous ways that a user can interact with a graphics system.
  - In a typical paint program it would look like:



■ 7.1 Application Programmer's Interfaces

- What is an API?



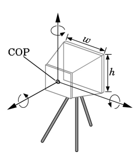
- Why do you want one?

- If we are to follow the synthetic camera model, we need functions in the API to specify:
  - Objects
  - Viewer
  - Light Sources
  - Material Properties

- Objects are usually defined by a set of vertices
  - The following code fragment defines a triangular polygon in OpenGL

```
– glBegin(GL_POLYGON);
–     glVertex3f(0.0,0.0,0.0);
–     glVertex3f(0.0,1.0,0.0);
–     glVertex3f(0.0,0.0,1.0);
– glEND();
```


- We can define a viewer or camera in a variety of ways



  - We can identify four types of necessary specifications:
    - Position
    - Orientation
    - Focal length
    - Film plane

- Light sources can be defined by their location, strength, color, and directionality.
  - API's provide a set of functions to specify these parameters for each source.
- Material properties are characteristics, or attributes, of the objects
  - such properties are usually defined through a series of function calls at the time that each object is defined.

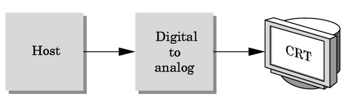
■ 7.2 A Sequence of Images

- In Chapter 2 , we begin our detailed discussion of the OpenGL API
- Color Plates 1 through 8 show what is possible with available hardware and a good API, but also they are not difficult to generate

8. Graphics Architectures

- On one side of the API is the application program. On the other is some combination of hardware and software that implements the functionality of the API
- Researchers have taken various approaches to developing architectures to support graphics APIs

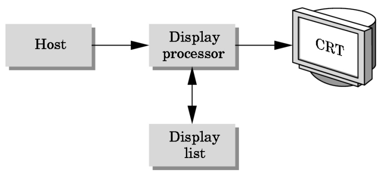
- Early systems used general purpose computers
  - single processing unit



  - In the early days of computer graphics, computers were so slow that refreshing even simple images, containing a few hundred line segments, would burden an expensive computer

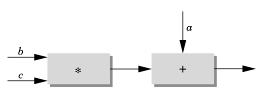
■ 8.1 Display Processors

- The earliest attempts to build a special purpose graphics system were concerned primarily with relieving the task of refreshing the display

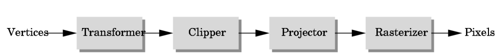


■ 8.2 Pipeline Architectures

- The major advances in graphics architectures parallel closely the advances in workstations.
- For computer graphics applications, the most important use of custom VLSI circuits has been in creating pipeline architectures
- A simple arithmetic pipeline is shown here:



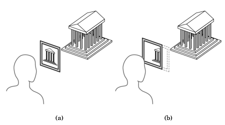
- If we think in terms of processing the geometry of our objects to obtain an image, we can use the following block diagram:



- We will discuss the details of these steps in subsequent chapters.

- 8.3 Transformations
  - Many of the steps in the imaging process can be viewed as transformations between representations of objects in different coordinate systems
    - for example: from the system in which the object was defined to the system of the camera
  - We can represent each change of coordinate systems by a matrix
    - We can represent successive changes by multiplying (or concatenating) the individual matrices into a single matrix.

- 8.4 Clipping
  - Why do we Clip?



  - Efficient clipping algorithms are developed in Chapter 7

- 8.5. Projection
  - In general three-dimensional objects are kept in three dimensions as long as possible, as they pass through the pipeline.
  - Eventually though, they must be projected into two-dimensional objects.
  - There are various projections that we can implement.
  - We shall see in Chapter 5 that we can implement this step using 4 x 4 matrices, and, thus, also fit it into the pipeline.

■ 8.6 Rasterization

- Finally, our projected objects must be represented as pixels in the frame buffer.
- We discuss this scan-conversion or rasterization process in Chapter 7

■ 8.7 Performance Characteristics

- There are two fundamentally different types of processing
  - Front end -- geometric processing, based on the processing of vertices
    - ideally suited for pipelining, and usually involves floating-point calculations.
    - The geometry engine developed by SGI was a VLSI implementation for many of these operations in a special purpose chip. These became the basis for a series of graphics workstations.
  - Back end -- involves direct manipulation of bits in the frame buffer.
    - Ideally suited for parallel bit processors.

9. Summary

- In this chapter we have set the stage for our top-down development of computer graphics.
  - Computer graphics is a method of image formation that should be related to classical methods -- in particular to cameras
  - Our next step is to explore the application side of Computer Graphics programming
    - We will use the OpenGL API

10. Suggested Readings

- Journals:
  - *Computer Graphics* -- ACM
  - *IEEE Compute Graphics and Applications*
- Textbooks:
  - Foley et.al.
  - Hearn and Baker
  - Hill

Exercises -- Due next class

- 1.1
- 1.7
- 1.11