Computer Graphics - Chapter 1
Graphics Systems and Models

Objectives are to learn about:
Applications of Computer Graphics
Graphics Systems
Images: Physical and Synthetic
The Human Visual System
The Pinhole Camera
The Synthetic-Camera Model
The Programmer’s Interface
Graphics Architectures

Applications of Computer Graphics
Applications of Computer Graphics are:
1. Display of information
   - Architectures - floor plans of buildings
   - Geographical information - maps
   - Medical diagnoses - medical images
   - Design engineers - chip layout
   - etc.
2. Design
   - There are two types of design problems: Overdetermined (posses no optimal solution) or underdetermined (have multiple solutions).
   - Thus, the design is an iterative process.
   - Computer-aided Design (CAD) tools assist architectures and designer of mechanical parts with their design.
Applications of Computer Graphics - cont

3. Simulations
   - Real-life examples: flight simulators, Command and conquer simulators, motion-pictures, virtual reality, and medical imaging.

4. User Interface
   - Interaction with the computer via visual paradigm that includes windows, icons, menus, and pointing devices.

A Graphics System
A computer graphics system with the following components:
   1. Processor
   2. Memory
   3. Frame buffer
   4. Output devices
   5. Input devices

A Typical Graphics System

Imaged formed here
**Pixels and the Frame Buffer**

All graphics systems are raster-based. A picture is produced as an array – the raster – of picture elements, pixels. Each pixel corresponds to a small area of the image.

Pixels are stored in a part of memory called the frame buffer.

A) Image of Yeti the cat.  
B) Detail of area around one eye showing individual pixels.

---

**A sample Image in Pixel_Gray_Map (PGM) Format**

```
P2
# test
4 4
255
0 50 127 255
255 200 127 0
10 60 100 250
20 70 80 200
```

This is a 4_by_4 8 bit image, i.e., 255 represents white and 0 represents black, other numbers are used for different shades of gray.
Pixels and the Frame Buffer-cont.
Frame buffers may be implemented with Video Random-access Memory (VRAM) or Dynamic Random-access Memory (DRAM).

The depth of the frame buffer, defines the number of bits that are used for each pixel and defines the number of colors.

In full-color (true color or RGB) systems, there are 24 (or more) bits per pixel. A 24 bit system can have up to 16,777,216 different colors.

The resolution is the number of pixels in the frame buffer and determines the detail that you can see in the image.

The conversion of geometric entities to pixel assignments in the frame buffer is known as rasterization, or scan conversion.

Output Devices
The dominant type of display used is cathode-ray tube (CRT).

How does it work?
- Electrons strike the phosphor coating on the tube, light gets emitted,
- The direction of the beam is controlled by the two pairs of deflection plates.
- The output of a computer, digital, is converted to voltage, analog, across the deflection plates using a digital-to-analog converter.

Since the beam can be moved directly from one position to any other position, sometimes this device is called random-scan or calligraphic CRT.
Output Devices – cont.

A typical CRT device emits light for only short time - a few milliseconds after the phosphor is excited by the electron beam.

- Human eyes can see a steady image on most CRT displays when the same path is retraced or refreshed by the beam at least 50 times per second (50 HZ).
- In a raster system, the graphics system takes pixels from the frame buffer and displays them as points on the surface of the display. The rate of display must be high enough to avoid flicker. This is known as the refresh rate.
- There are two fundamental ways that pixels are displayed:
  1. **noninterlaced**: pixels are displayed row-by-row, at refresh rate of 50-85 times per second.
  2. **interlaced**: odd rows and even rows are refreshed alternately. For a system operating at 60 HZ, the entire display is redrawn 30 times per second.

Output Devices – cont.

A typical color CRT have three different colored phosphors (red, green, and blue) arranged in small groups.

A common style arranges the phosphors in triangular groups called **triads**, each consisting of three phosphors (RGB).

The **shadow mask** ensures that an electron beam excites only phosphors of the same color.
Input Devices

Most graphics systems provide a keyboard and at least one other input device. The most common devices are the mouse, the joystick, and the data tablet.

Often called pointing devices, these devices allow a user to indicate a particular location on the display.

Images: Physical and Synthetic

A computer generated image is a synthetic or artificial, in the sense that the object being imaged does not exist physically.

In order to understand how synthetic images are generated, we first look into the ways traditional imaging systems such as cameras form images.

Objects and Viewers

Our world is the world of three dimensional (3-D) objects. We refer to a location of a point on an object in terms of some convenience reference coordinate system.

There is a fundamental link between the physics and the mathematics of image formation. We exploit this link in our development of computer image formation.

There are two entities that are part of image formation: object and viewer. The object exists in space independent of any image-formation process, and of any viewer.

In computer graphics, we form objects by specifying the positions in space of various geometric primitives, such as points, lines, and polygons.

In most graphics systems, a set of locations in space, or of vertices, is sufficient to define, or approximate, most objects.

For example: A line can be defined by two vertices.
How many circles do you see on the screen?

How many circles are there on the screen?

Objects and Viewers – cont.

To form an image, we must have someone, or something, that is viewing our objects, be it a human, a camera, or a digitizer. It is the viewer that forms the image of our objects.

In human visual system, the image is formed on the back of the eye, on the retina. In a camera, the image is formed on the film plane.

An image may be confused with an object.
How many circles are there on the screen?

Or you could see: |||||

**Objects and Viewers – cont.**

The object and the viewer exist in a three-dimensional world. The image that they define, what we find on the film, is two-dimensional.

The process by which the specification of the object is combined with the specification of the viewer to produce a two-dimensional image is the essence of image formation.
Light and Images

In our previous discussion of image-formation, we didn’t talk about light. If there is no light source, an object would be dark and there won’t be anything visible of the image.

Light usually strikes various parts of the object and a portion of the reflected light will enter the camera through the lens.

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

Light is formed of electromagnetic radiation characterized by its wavelength or frequency.

Light and Images – cont.

The electromagnetic spectrum includes radio waves, infrared (heat), and a portion that causes a response in our visual systems, visible light spectrum.

The color of light source is determined by the energy that it emits at various wavelengths. In graphics, we use the geometric optics which models light sources as emitters of light energy that have a fixed rate or intensity.

An ideal point source emits energy from a single location at one or more frequencies equally in all directions. We also assume purely monochromatic lighting - a source of a single frequency.
Ray Tracing

We can model an image by following light from a source. The light that reaches the viewer, will determine the image-formation.

A ray is a semi-infinite line that emanates from a point and travels to infinity in a particular direction.

Light travels in straight line, thus only a portion the light will get to the viewer.

The light from the light source can interact with the surface of the object differently, depending on the orientation and the luminosity of the surface.

A diffuse surface scatters light in all directions while a transparent surface passes light through which may result in the light being bent or refracted.

Ray Tracing – cont.

Ray tracing is an image-formation technique used as the basis for producing computer-generated images.

The ray tracing model can be used to simulate physical effects as complex as we wish. Bare in mind that complexity will cost the computing time and resources.

Although, ray tracing presents a close approximation to the physical world, it is not well suited for fast computation.

We simplify our objects by assuming that all our objects are equally bright.

Why do we need such an assumption?
The Human Visual System

Lights enters the eye through Cornea and Lens.

The Iris opens and closes to adjust the amount of light.

The lens forms an image on a 2-D structure called Retina.

There are rods and cones that Act like light sensors on the Retina.

Rods are low-level light sensors, night vision.

Cones are responsible for our day vision.

The Human Visual System – cont.

The sensors in the human eye do not react uniformly to light energy at different wavelengths. There are three types of cones and a single type of rod.

Intensity is a physical measure of light energy, brightness is a measure of how intense we perceive the light emitted from an object to be.

How it is vs. how we see it …

Relative sensitivity of human visual system CIE standard observer curve.

Most sensitive to: Green
Least sensitive: Red & Yellow

Human color-vision capabilities are due to the different sensitivities of the three types of cones. One is centered in the blue range, one in the green, and one in the yellow.
The Pinhole Camera

Projection Point of \((x,y,z)\)

Projection Point of \((y,z)\)

The field or angle view of our camera is the angle made by the largest object that the camera can image on its film plane. It can be computed using:

\[
\tan(\theta) = \frac{h}{2d}
\]

\[
\theta = 2 \tan^{-1} \left( \frac{h}{2d} \right)
\]

The ideal pinhole camera has an infinite depth of field. This camera has two disadvantages:
1) the pinhole is very small, it admits only a single ray from a point source,
2) the camera cannot be adjusted to have a different angle of view.

For our purposes, we work with a pinhole camera whose focal length is the distance \(d\) from the front of the camera to the film plane. Like the pinhole camera, computer graphics produces images in which all objects are in focus.
The Synthetic – Camera Model

We look at creating a computer-generated image as being similar to forming an image using an optical system. This paradigm is known as synthetic-camera model.

The image is formed on the back of the camera, so we can emulate this process to create artificial images.

Some basics principals:

1) The specification of the objects is independent of the specification of the viewer. Expect separate functions for specifying the objects and the viewers in the graphic library that you will use.

An imaging system:
- a viewer (bellows camera)
- an object

The Synthetic – Camera Model -cont.

2) We can compute the image using simple trigonometric calculations.

We find the image of a point on the object by drawing a line, projector, from the point to the center of the lens, or the center of projection. In our synthetic camera, the film plane is moved in front of the lens and is called Projection plane.
The Synthetic – Camera Model -cont.

We must consider the limited size of the image. Not all objects can be imaged onto the pinhole camera’s film plane.

The angle of view expresses this limitation.

In our synthetic camera, we place this limitation by placing a clipping rectangle or clipping window in the projection plane.

Given the location of the center of the projection, the location and orientation of the projection plane, and the size of the clipping rectangle, we can determine which objects will appear in the image.
The Programmer’s Interface

There are many sophisticated commercial software products with nice graphical interfaces. Every one of them has been developed by someone.

Some of us still need to develop graphics applications to interface with these sophisticated software products.
Application Programmer’s Interfaces

The interface between an application program and a graphics system can be specified through a set of functions that resides in a graphics library. These specifications are called the application programmer’s interface (API).

The synthetic-camera model is the basis for a number of popular APIs, including OpenGL, PHIGS, Direct3D, VRML, and JAVA-3D. In order to model synthetic-camera model we need functions in the API to specify:
- Objects
- Viewer
- Light sources
- Material properties

The Programmer’s Interface – cont.

Objects are defined by sets of vertices. For simple objects, line, rectangle, and polygon, there is a simple relationship between a list of vertices and the object.

For more complex objects, there may be multiple ways of defining the object from a set of vertices.

Most APIs provide similar sets of primitive objects for the user. OpenGL defines primitives objects through lists of vertices.

Here is an example of how a triangular polygon is defined:

```c
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();
```

There are five function calls in this code segment.

Create a rectangle of size 2 by 1. The top left (1st point) is still at (0, 0, 0).
The Programmer’s Interface – cont.

We can define a viewer or camera in a variety of ways. Looking at the camera given here, we can identify four types of necessary specifications:

1. **Position**: The camera location usually is given by the position of the center of the lens (center of the projection).

2. **Orientation**: Once we have positioned the camera, we can place a camera coordinate system with its origin at the center of projection. We can then rotate the camera independently around the three axes of this system.

3. **Focal length**: The focal length of the lens determines the size of the image on the film plane or, equivalently, the portion of the world the camera sees.

4. **Film plane**: The back of the camera has a height, $h$, and width, $w$, on the bellows camera, and in some APIs, the orientation of the back of the camera can be adjusted independently of the orientation of the lens.
To develop the specifications for the camera location and orientation, one can use a series of coordinate system transformations. These transformations convert object positions represented in the coordinate system that specifies object vertices to object positions in a coordinate system centered at the center of projection. This may require setting and adjusting many parameters which will make it difficult to get a desired image. Part of the problem lies with the synthetic-camera model. Classical viewing techniques stress the relationship between object and the viewer, rather than the independence that the synthetic-camera model emphasizes.

The Programmer’s Interface – cont.

OpenGL API allows us to set transformations with complete freedom.

```
gluLookAt(cop_x, cop_y, cop_z, at_x, at_y, at_z, ...);
```

This function call points the camera from a center of projection toward a desired point.

```
gluPerspective(field_of_view, ...);
```

This function selects a lens for a perspective view.

A light source can be defined by its location, strength, color, and directionality. APIs provide a set of functions to specify these parameters for each source. Material properties are the attributes of an object. Such properties are defined through a series of functions. Both the light sources and material properties depend on the models of light-material interactions supported by the API.
**Sequence of Images and the Modeling-Rendering Paradigm**

OpenGL allows you to write graphical application programs. The images defined by your programs will be formed automatically by the hardware and software implementation of the image-formation process.

Sometimes a scene consists of several objects. Although, programmer may have used sophisticated data structures to model each object and the relationship among the objects, the rendered scene shows only the outlines of the objects. This type of image is known as a **wireframe** image, because we can see only the edge of surfaces.

A common approach in developing realistic images is to separate the modeling of the scene from the production of the image – the **rendering** of the scene.

Thus, the modeler and renderer are done with different software and hardware.

---

**Graphics Architectures**

On one side of the API is the application program. On the other is some combinations of hardware and software that implements the functionality of the API.

The advances in graphics architectures parallel closely the advances in workstations.

Special-purpose VLSI circuits have significantly improved the graphics technology.
Graphics Architectures – cont.

The most important use of custom VLSI circuits has been in creating pipeline architectures.

This architecture may not make any difference when computing a single multiplication and addition, but it will make a significant difference when we do this operations for many sets.

A set of primitives are defined by a set of vertices. Because our representation is given in terms of locations in space, we can refer to the set of primitive types and vertices as the geometry of the data.

The four major steps in imaging process are:

OpenGL Libraries

• OpenGL core library
  – OpenGL32 on Windows
  – GL on most unix/linux systems (libGL.a)

• OpenGL Utility Library (GLU)
  – Provides functionality in OpenGL core but avoids having to rewrite code

• Links with window system
  – GLX for X window systems
  – WGL for Windows
  – AGL for Macintosh
GLUT

- OpenGL Utility Toolkit (GLUT)
  - Provides functionality common to all window systems
    - Open a window
    - Get input from mouse and keyboard
    - Menus
    - Event-driven
  - Code is portable but GLUT lacks the functionality of a good toolkit for a specific platform
    - No slide bars