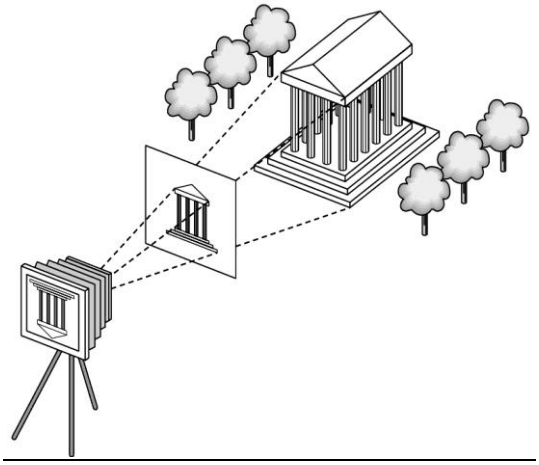
1.5 The Synthetic camera model



The paradigm which looks at creating a computer generated image as being similar to forming an image using an optical system.

Various notions in the model :

Center of Projection

Projector lines

Image plane

Clipping window

- In case of image formation using optical systems, the image is flipped relative to the object.
- In synthetic camera model this is avoided by introducing a plane in front of the lens which is called the image plane.

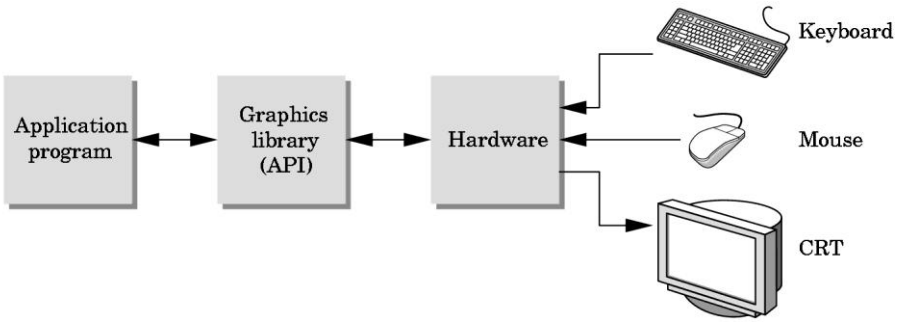
The angle of view of the camera poses a restriction on the part of the object which can be viewed.

This limitation is moved to the front of the camera by placing a Clipping Window in the projection plane.

1.6 Programer's interface :

A user interacts with the graphics system with self-contained packages and input devices. E.g. A paint editor.

This package or interface enables the user to create or modify images without having to write programs. The interface consists of a set of functions (API) that resides in a graphics library



- The application programmer uses the API functions and is shielded from the details of its implementation.

- The device driver is responsible to interpret the output of the API and converting it into a form understood by the particular hardware.

- The pen-plotter model

This is a 2-D system which moves a pen to draw images in 2 orthogonal directions.

E.g. : LOGO language implements this system.

moveto(x,y) – moves pen to (x,y) without tracing a line.

lineto(x,y) – moves pen to (x,y) by tracing a line.

Alternate raster based 2-D model :

Writes pixels directly to frame buffer

E.g. : write_pixel(x,y,color)

In order to obtain images of objects close to the real world, we need 3-D object model.

3-D APIs (OpenGL - basics)

To follow the synthetic camera model discussed earlier, the API should support:

Objects, viewers, light sources, material properties.

OpenGL defines primitives through a list of vertices.

Primitives: simple geometric objects having a simple relation between a list of vertices

Simple prog to draw a triangular polygon :

```
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( );

Specifying viewer or camera:

Position - position of the COP

Orientation – rotation of the camera along 3 axes

Focal length – determines the size of image

Film Plane – has a height & width & can be adjusted independent of orientation of lens.

Function call for camera orientation :

gluLookAt(cop_x,cop_y,cop_z,at_x,at_y,at_z,up_x,up_y,up_z);

gluPerspective(field_of_view,aspect_ratio,near,far);

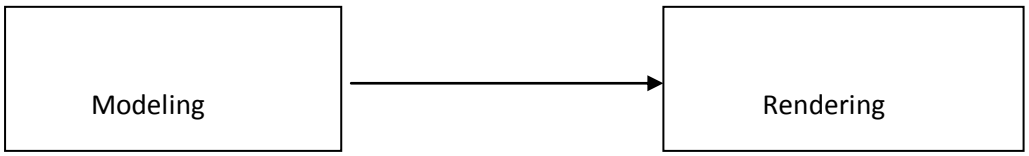
Lights and materials :

- Types of lights
  - Point sources vs distributed sources
  - Spot lights
  - Near and far sources
  - Color properties

- Material properties
  - Absorption: color properties
  - Scattering

Modeling Rendering Paradigm :

Viewing image formation as a 2 step process



E.g. Producing a single frame in an animation:

1st step : Designing and positioning objects

2nd step : Adding effects, light sources and other details

The interface can be a file with the model and additional info for final rendering.