**Display of Information**

* Classical graphics techniques arose as a medium to convey information among people.
* We have computer plotting packages that provide a variety of plotting techniques and color tools that can handle multiple large data sets.
* The field of information visualization is becoming increasingly more important as we have to deal with understanding complex phenomena from problems in bioinformatics to detecting security threats.

**Design**

* Professional such as engineering and architecture are concerned with design.
* The use of interactive graphical tools in computer-aided design (CAD) pervades fields including as architecture, mechanical engineering, the design of very-large-scale integrated (VLSI) circuits, and the creation of characters for animations.

**Simulation and Animation**

* Once graphics systems evolved to be capable of generating sophisticated images in real time, engineers and researchers began to use them as simulators.
* One of the most important uses has been in the training of pilots. Graphical flight simulators have proved to increase safety and to reduce training expenses. The use of special VLSI chips has led to a generation of arcade games s as sophisticated as flight simulators.

**User Interfaces**

* Interaction with computers has become dominated by a visual paradigm that includes windows, icons, menus, and plotting device, such as mouse.
* User interfaces demonstrate the variety of the tools available in high level modeling packages and the interactive devices the user can employ in modeling geometric object.

**1.1.1 Image Types**

* **2D computer graphics:**

2D computer graphics are the computer-based generation of digital images mostly from two-dimensional models, such as 2D geometric models, text, and digital images, and by techniques specific to them.2D computer graphics are mainly used in applications that were originally developed upon traditional printing and drawing technologies, such as typography, cartography, technical drawing, advertising. Two-dimensional models are preferred, because they give more direct control of the image than 3D computer graphics, whose approach is more akin to photography than to typography.

There are two approaches to 2D graphics: vector and raster graphics.

* Pixel art: Pixel art is a form of digital art, created through the use of raster graphics software, where images are edited on the pixel level.
* Vector graphics: Vector graphics formats are complementary to raster graphics, which is the representation of images as an array of pixels, as it is typically used for the representation of photographic images.
* **3D computer graphics:**

With the birth of the workstation computers(like LISP machines,paintbox computers and Silicon Graphics workstations)came the 3D computer graphics.3D computer graphics in contrast to 2D computer graphics are graphics that use a three-dimensional representation of geometric data that is stored in the computer for the purposes of performing calculations and rendering 2D images.

**Some major advances in 3D computer graphics since then have been:**

* Flat shading: A technique that shades each polygon of an object based on the polygon's "normal" and the position and intensity of a light source.
* Gouraud shading: Invented by Henri Gouraud in 1971, a fast and resource-conscious technique used to simulate smoothly shaded surfaces by interpolating vertex colors across a polygon's surface.
* Texture mapping: A technique for simulating surface detail by mapping images(textures) onto polygons.
* Phong shading: Invented by Bui Toc Phong, a smooth shading technique that approximates curved-surface lighting by interpolating the vertex normals of a polygon across the surface; the lighting model includes glossy reflection with a controlable level of gloss.
* Bump mapping: Invented by Jim Blinn, a normal-perturbation technique used to simulate bumpy or wrinkled surfaces.
* Raytracing: A method based on the physical principles of geometric optics that can simulate multiple reflections and transparency.
* Radiosity: a technique for global illumination that uses radiative transfer theory to simulate indirect (reflected) illumination.

**1.1.2Applications of computer graphics**

Some of the applications of computer graphics are listed below:

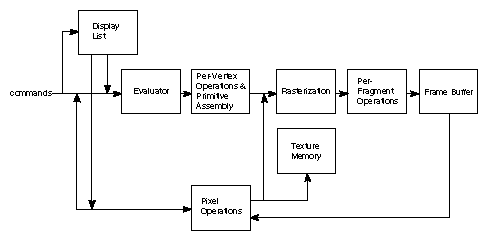
* Computational biology
* Computational physics
* Computer-aided design
* Computer simulation
* Digital art
* Education
* Graphic design
* Video Games
* Virtual reality
* Web design

**1.1.3 OpenGL**

As a software interface for graphics hardware, OpenGL's main purpose is to render two- and three-dimensional objects into a frame buffer. These objects are described as sequences of vertices (which define geometric objects) or pixels (which define images). OpenGL performs several processing steps on this data to convert it to pixels to form the final desired image in the frame buffer.

## Basic OpenGL Operation

The figure shown below gives an abstract, high-level block diagram of how OpenGL processes data. In the diagram, commands enter from the left and proceed through what can be thought of as a processing pipeline. Some commands specify geometric objects to be drawn, and others control how the objects are handled during the various processing stages.



**Figure 1.1 OpenGL Block Diagram**

As shown by the first block in the diagram, rather than having all commands proceeds immediately through the pipeline, you can choose to accumulate some of them in a *display list* for processing at a later time. The *evaluator* stage of processing provides an efficient means for approximating curve and surface geometry by evaluating polynomial commands of input values. During the next stage, *per-vertex operations and primitive assembly*, OpenGL processes geometric primitives—points, line segments, and polygons, all of which are described by vertices. Vertices are transformed and lit, and primitives are clipped to the viewport in preparation for the next stage.*Rasterization* produces a series of frame buffer addresses and associated values using a two-dimensional description of a point, line segment, or polygon.

Each *fragment* so produced is fed into the last stage, *a per-fragment operation, which performs* the final operations on the data before it's stored as pixels in the *frame buffer*. These operations include conditional updates to the frame buffer based on incoming and previously stored z-values (for z-buffering) and blending of incoming pixel colors with stored colors, as well as masking and other logical operations on pixel values.

Input data can be in the form of pixels rather than vertices. Such data, which might describe an image for use in texture mapping, skips the first stage of processing described above and instead is processed as pixels, in the *pixel operations* stage. The result of this stage is either stored as *texture memory*, for use in the rasterization stage, or rasterized and the resulting fragments merged into the frame buffer just as if they were generated from geometric data.All elements of OpenGL state, including the contents of the texture memory and even of the frame buffer, can be obtained by an OpenGL application.

**1.1.4GLUT**

GLUT is the OpenGL Utility Toolkit, a window system independent toolkit for writing OpenGL programs. It implements a simple windowing application programming interface (API) for OpenGL. GLUT makes it considerably easier to learn about and explore OpenGL programming. GLUT provides a portable API so you can write a single OpenGL program that works on both Win32 PCs and X11 workstations.

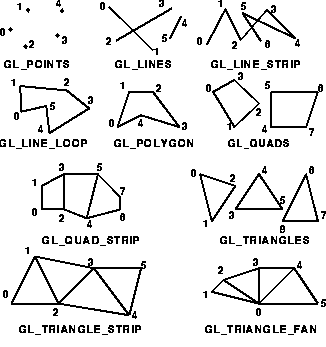
GLUT is designed for constructing small to medium sized OpenGL programs. While GLUT is well-suited to learning OpenGL and developing simple OpenGL applications, GLUT is not a full-featured toolkit so large applications requiring sophisticated user interfaces are better off using native window system toolkits like Motif. GLUT is simple, easy, and small. My intent is to keep GLUT that way.

**The GLUT library supports the following functionality:**

* Multiple windows for OpenGL rendering.
* Call back driven event processing.
* An `idle' routine and timers.
* Utility routines to generate various solid and wire frame objects.
* Support for bitmap and stroke fonts.
* Miscellaneous window management functions.

# OpenGL Primitives

# The programmer is provided the following primitives for use in constructing geometric objects.



**1.2 Overview of the project**

In this project we designed the simulation of windmill using OpenGL. We used transformation functions like translate and rotate functions to design blades of the windmill. We used many OpenGL inbuilt function to design the structure of windmill.

This projectconsist of many user defined function such as increasing windmill fan speed, decreasing windmill fan speed, side views, front and back views, custom angle of rotation of entire windmill structure.

It provides several options which can be interacted through menus. The user can also interact with program through mouse, keyboard functions. The options provided by the menu are views like side view, back view, front view, custom view. Using mouse, if we click left side it rotates to left and on successive clicking speed increases, if we click right button speed decreases and on successive clicking, it turns rotating towards right and vice versa.

We can rotate the entire windmill with respect to its axis using the arrow keys of keyboard. It can be rotate through 3600

**1.3 Aim of the project**

The aim of the project is to develop a suitable graphics package to simulate the Windmill and to implement the skills learned in Interactive Computer Graphics and Visualization theory, using OpenGL.