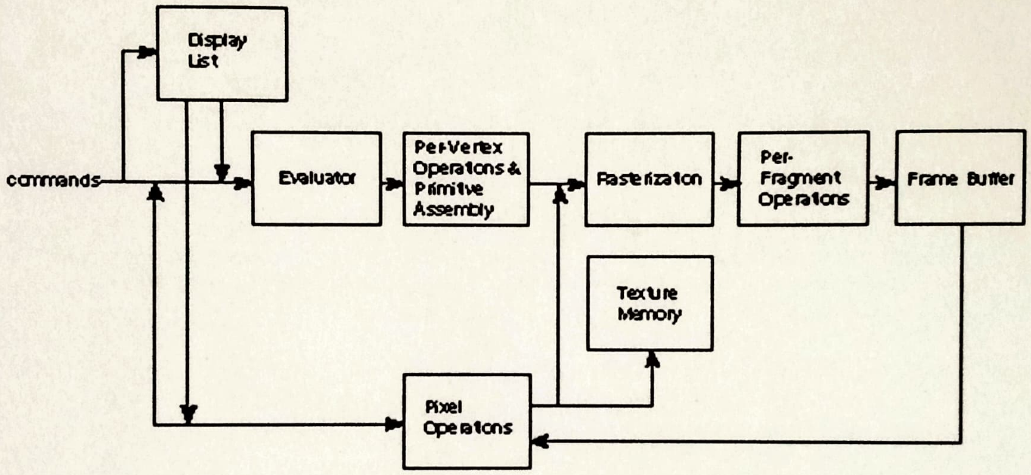
Experiment 1

Introduction to OPENGL

➢ What is OPENGL?

As a product interface for designs equipment, OpenGL's principle intention is to render two- and three-dimensional articles into an edge cradle. These articles are portrayed as arrangements of vertices (which characterize geometric items) or pixels (which characterize pictures). OpenGL plays out a few preparing ventures on this information to change over it to pixels to shape the last wanted picture in the edge cushion.

OpenGL Block Diagram.



➢ What is glu/glut?

GLU (OpenGL Utility Library) is a graphics library for OpenGL, comprising of utility functions which can be implemented with OpenGL. The functions mainly focus on primitive rendering and mapping between screen- and world-coordinates, etc. GLUT (OpenGL Utility Toolkit) is a library of utilities for OpenGL, which primarily focuses on window definition, window control and monitoring of keyboard and mouse input.

- OpenGL v2.1 (latest) is the "core" library that is platform independent
- GLUT v3.7 is an auxiliary library that handles window creation, OS system calls (mouse buttons, movement, keyboard, etc), callbacks.
- GLU is an auxiliary library that handles a variety of graphics accessory functions

The GLU library was developed to provide both useful functions encapsulate the basic OpenGL commands and complex components supporting advanced rendering techniques, similar to the MFC in Windows programming. Apprently GLU is targeted on the first disadvantage of the basic OpenGL library we mentioned above.

The GLUT library, on the other hand, was developed to provide a platform-independent interface to the windowing system and input devices, thus targeted on the second disadvantage of the basic OpenGL library. The implementation of the GLUT library is of course platform-dependent, but the interface it provides to the programmers is platform-independent. GLUT is quite good for small programs such as as demos, but is usually not powerful and flexible enough to support real applications.

➢ Opengl Architechture

CPU-GPU Cooperation

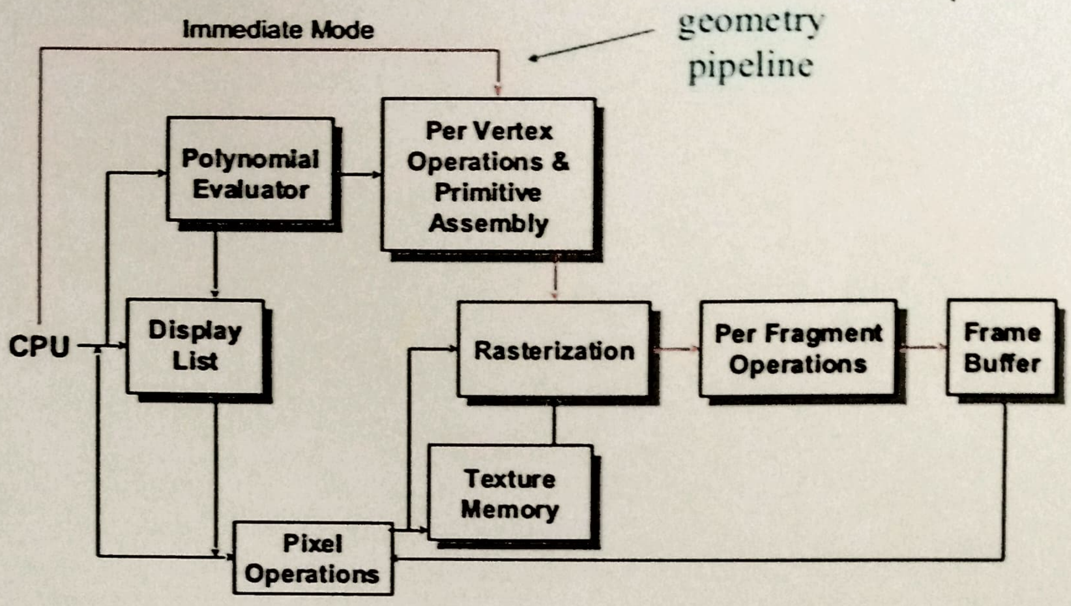
The architecture of OpenGL is based on a client-server model. An application program written to use the OpenGL API is the "client" and runs on the CPU. The implementation of the OpenGL graphics engine (including the GLSL shader programs you will write) is the "server" and runs on the GPU. Geometry and many other types of attributes are stored in buffers called Vertex Buffer Objects (or VBOs). These buffers are allocated on the GPU and filled by your CPU program. We will get our first glimpse into this process (including how these buffers are allocated, used, and deleted) in the first sample program we will study.

Modeling, rendering, and interaction is very much a cooperative process between the CPU client program and the GPU server programs written in GLSL. An important part of the design process is to decide how best to divide the work and how best to package and communicate required information from the CPU to the GPU. There is no standard "best way" to do this that is applicable to all programs, but we will study a few very common approaches.

Window Manager Interfaces

OpenGL is a pure output-oriented modeling and rendering API. It has no facilities for creating and managing windows, obtaining runtime events, or any other such window system dependent operation. OpenGL implicitly assumes a window-system interface that fills these needs and invokes user-written event handlers as appropriate.

OpenGL Architecture



➢ **Setting Up The Environment**