Graphical Languages

Unit 1

Graphics Software

- There are two general classification for graphics:
- General Programming Packages
- Special Purpose Application Packages

General Programming Packages

- It provides an extensive set of graphics functions that can be in high-level programming languages, such as C or FORTAN.
- Example: GL (Graphics Library) system on Silicon Graphics Equipment, which includes basic functions for generating picture components (straight lines, polygon , circles, and other figures), setting colors and intensity values, selecting views, and applying transformations.

Special Purpose Application Packages

- Designed for non-programmers, so that users can generate displays without worrying about how graphics operation work.
- The interface to the graphics routines in such packages allows users to communicate with the programs in their own terms.
- For examples: the artist's painting programs and various business, medical, and Computer-Aided-Design (CAD) system.

Graphics software available in market

There are many graphic software available in market. They can be categorized as:

1. **Paint Program**: Paint program works with bit map images
2. **Photo manipulating program**: It works with bit map images and is widely used to edit digitized photographs.
3. **Computer Aided Design Program** : CAD software is used in technical design fields to create models of objects that will be built or manufactured CAD software allows user to design objects in 3 dimensions and can produce 3-D frames and solid models.
4. **3-D Modeling Programs**: It is used to create visual effects. 3-D modeling program works by creating objects like surface, solid, polygon etc.
5. **Animation** : Computer are used to create animation for use in various fields, including games, and movies, composing to allow game makers and film makers to add characters and objects to scenes that din not original contain them.

Software Standards

- The primary goal of standardized graphics software is *portability*.
- The Software Standards and package are designed with standard graphics functions, software can be moved easily from one hardware to another and used in different implementations and applications.
- Without standards, programs designed for one hardware system often cannot be transferred to another system without extensive rewriting of the programs.
- Some Software Standards are:
- GKS
- PHIGS
- PHIGS+
- OpenGL

Software Standards

- The International Standards Organization (**ISO**) and American Nation Standards Institute (**ANSI**) adopted General Kernel System (**GKS**) as the first software standard.
- **PHIGS** (Programmer's Hierarchical Interactive Graphics System), which is an extension of GKS, increased capabilities for object modeling, color specifications, surface rendering, and picture manipulations.
- **PHIGS+** was developed to provide 3-D surface shading capabilities not available in PHIGS.

OpenGL

- **OpenGL** (Open Graphics Library) is a standard specification defining a cross-language cross platform API for writing applications that produce 2D and 3D computer graphics.
- The interface consists of over 250 different function calls, which can be used to draw complex three-dimensional scenes from simple primitives.
- OpenGL was developed by Silicon Graphics Inc. (SGI) in 1992 and is widely used in CAD, virtual reality, scientific visualization, information visualization, and flight simulation.
- It is also used in video games, where it competes with Direct3D on Microsoft Windows Platforms.
- OpenGL serves two main purposes:

  - To hide the complexities of interfacing with different 3D accelerators, by presenting the programmer with a single, uniform API.

  - To hide the different capabilities of hardware platforms, by requiring that all implementations support the full OpenGL feature set (using software emulation if necessary)

OpenGL

- Many games also use OpenGL for their core graphics-rendering engines.
- Note that OpenGL is just a graphics library; unlike DirectX, it does not include support for sound, input, networking, or anything else not directly related to graphics.
- The following list briefly describes the major graphics operations that OpenGL performs to render an image on the screen:
  i. Construct shapes from geometric primitives (points, lines, polygons, images and bitmaps).
  ii. Arrange the objects in 3-D space and select the desired vantage point for viewing the compose scene.
  iii. Calculate the colors of all the objects.
  iv. Convert the mathematical descriptions of objects and their association color information to pixels on the screen. This process is called *rasterization*.

OpenGL Rendering Pipeline

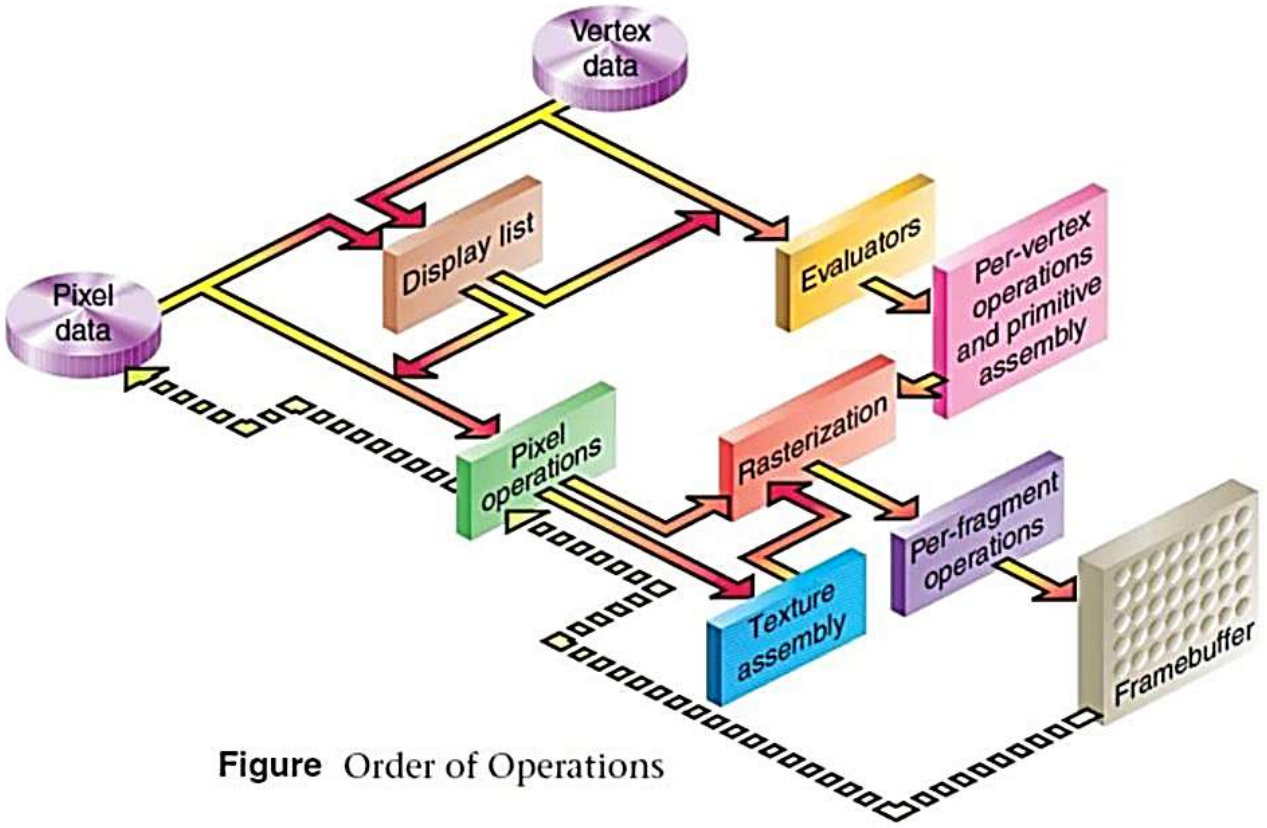

Figure Order of Operations

OpenGL Rendering Pipeline

Display Lists :

- All data, whether it describes geometry or pixels, can be saved in a display list for current or later use.

Evaluators :

- All geometric primitives are eventually described by vertices.
- Parametric curves and surfaces may be initially described by control points and polynomial functions called *basis functions*.
- Evaluators provide a method for deriving the vertices used to represent the surface from the control points. The method is *polynomial mapping*, which can produce surface normal, texture coordinates, colors, and spatial coordinates values from the control points.

OpenGL Rendering Pipeline

Per-Vertex Operations :

- For vertex data, next is the "pre-vertex operation" stage, which converts the vertices into primitives. Some types of vertex data (for example, spatial coordinates) are transformed by 4 x 4 floating-point matrices. Spatial Coordinates are projected from a position in the 3D world to a position on your screen.

Primitive Assembly :

- Clipping, a major part of primitive assembly, is the elimination of portions of geometry that fall outside a half-space, defined by a plane.
- The results of this stage are complete geometric primitives, which are the transformed and clipped vertices with related color, depth, and sometimes texture coordinates values and guidelines for the rasterization step.

OpenGL Rendering Pipeline

Pixel Operations :

- While geometric data takes one path through the OpenGL rendering pipeline, pixel data takes a different route.
- Pixels from an array in system memory are first unpacked from one of a variety of formats into the proper number of components.
- Next the data is scaled, biased, and processed by a pixel map. The results are clamped and then either written into texture memory or sent to the rasterization step.

Texture Assembly :

- OpenGL applications can apply texture images to geometric objects to make the objects look more realistic, which is one of the numerous techniques enabled by texture mapping.
- If several texture images are used, it's wise to put them into texture objects so that you can easily switch among them.

OpenGL Rendering Pipeline

Rasterization :

- Rasterization is the conversion of both geometric and pixel data into fragments. Each fragments square corresponds to a pixel in the frame buffer.
- Line and polygon stipples, line width, point size, shading model, and coverage calculations to support anti-aliasing are taken into consideration as vertices are connected into lines or the interior pixels are calculated for a filled polygon.
- Color and depth values are generated for each fragment square.

OpenGL Rendering Pipeline

Fragment Operations :

- Before values are actually stored in the frame buffer, a series of operations are performed that may alter or even throw out fragments. All these operations can be enable or disabled.
- The first operation that a fragment might encounter is texturing, where texel (texture element) is generated from texture memory for each fragment and applied to the fragment.
- Next, primary and secondary colors are combined, and a fog calculation may be applied. If your application is employing fragment shaders, the preceding three operations may be done in shader.
- After the final color and depth generation of previous operations, the scissor test, the alpha test, the stencil test, and the depth-buffer test are evaluated, if enabled. Filing an enabled test may end the continued processing of a fragments' square. Then, blending, dithering, logical operation, and masking by a bitmask may be performed.
- Finally, the thoroughly processed fragment is drawn into the appropriate buffer, where it has finally become a pixel and achieved its final resting place.

OpenGL Rendering Pipeline