**1. INTRODUCTION TO COMPUTER GRAPHICS**

* 1. **Computer graphics**

Graphics created using computers and the representation of image data by a computer specifically with help from specialized graphic hardware and software. The interaction and understanding of computers and interpretation of data has been made easier because of computer graphics. A computer graphic development has had a significant impact on many types of media and has revolutionized animation, movies and the video game industry.

Typically, the term *computer graphics* refers to several different things:

* the representation and manipulation of image data by a computer
* the various technologies used to create and manipulate images
* the sub-field of computer science which studies methods for digitally synthesizing and manipulating visual content.

Computer generated imagery can be categorized into several different types: two dimensional (2D),three dimensional (3D), and animated graphics. As technology has improved, 3D computer graphics have become more common, but 2D computer graphics are still widely used. Computer graphics has emerged as a sub-field of computer science which studies methods for digitally synthesizing and manipulating visual content. Over the past decade, other specialized fields have been developed like information visualization, and scientific visualization more concerned with "the visualization of three dimensional phenomena (architectural, meteorological, medical, biological, etc.), where the emphasis is on realistic renderings of volumes, surfaces, illumination sources, and so forth, perhaps with a dynamic (time) component".

**1.2 OpenGL**

**OpenGL** (**Open G**raphics **L**ibrary) is a cross-language, multi-platform API for rendering 2D and 3D computer graphics. The API is typically used to interact with a GPU, to achieve hardware-accelerated rendering.

OpenGL is designed as a streamlined, hardware-independent interface to be implemented on many different hardware platforms. To achieve these qualities, no commands for performing windowing tasks or obtaining user input are included in OpenGL; instead, we must work through whatever windowing system controls the particular hardware we’re using. Similarly, OpenGL doesn’t provide high-level commands for describing models of three-dimensional objects. Such commands might allow us to specify relatively complicated shapes such as automobiles, parts of the body, airplanes, or molecules.

With OpenGL, we must build up our desired model from a small set of *geometric primitives* - points, lines, and polygons. A sophisticated library that provides these features could certainly be built on top of OpenGL. The OpenGL Utility Library (GLU) provides many of the modeling features, such as quadric surfaces and NURBS curves and surfaces. GLU is a standard part of every OpenGL implementation.

In OpenGL, Rendering is the process by which a computer creates images from models. These *models*, or objects, are constructed from geometric primitives - points, lines, and polygons - that are specified by their vertices. The final rendered image consists of pixels drawn on the screen; a pixel is the smallest visible element the display hardware can put on the screen. Information about the pixels (for instance, what color they’re supposed to be) is organized in memory into bitplanes. A bitplane is an area of memory that holds one bit of information for every pixel on the screen; the bit might indicate how red a particular pixel is supposed to be, for example. The bitplanes are themselves organized into a *framebuffer*, which holds all the information that the graphics display needs to control the color and intensity of all the pixels on the screen.

**1.3 GLUT**

The OpenGL Utility Toolkit (GLUT) is a library of utilities for OpenGL programs, which primarily perform system-level I/O with the host operating system. Functions performed include window definition, window control, and monitoring of keyboard and mouse input. Routines for drawing a number of geometric primitives (both in solid and wireframe mode) are also provided, including cubes, spheres and the Utah teapot. GLUT also has some limited support for creating pop-up menus.

The two aims of GLUT are to allow the creation of rather portable code between operating systems (GLUT is cross-platform) and to make learning OpenGL easier. Getting started with OpenGL while using GLUT often takes only a few lines of code and does not require knowledge of operating system–specific windowing APIs.

The GLUT library supports the following functionality:

* Multiple windows for OpenGL rendering.
* Callback 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.

**1.4 SOIL**

SOIL (Simple OpenGL Image Library) is a tiny C library used primarily for uploading textures into OpenGL. It is based on stb\_image version 1.16, the public domain code from Sean Barrett. It can load TGA and DDS files, and perform common functions needed in loading OpenGL textures. SOIL can also be used to save and load images in a variety of formats (useful for loading height maps, non-OpenGL applications, etc.)

You can start using it in your project by linking with SOIL and adding the src directory to your include path. Although SOIL includes functions to automatically create a texture from an image, it uses features that aren't available in modern OpenGL. Because of this we'll simply use SOIL as image loader and create the texture ourselves.

**Features:**

1. **Readable Image Formats:**

* BMP - non-1bpp, non-RLE (from stb\_image documentation)
* PNG - non-interlaced (from stb\_image documentation)
* JPG - JPEG baseline (from stb\_image documentation)

1. **Writeable Image Formats:**

* TGA - Greyscale or RGB or RGBA, uncompressed
* BMP - RGB, uncompressed
* DDS - RGB as DXT1, or RGBA as DXT5

1. **Can load an image file directly into a 2D OpenGL texture, optionally performing the following functions:**

* Can generate a new texture handle, or reuse one specified
* Can automatically rescale the image to the next largest power-of-two size
* Can automatically create MIPmaps
* Can scale the RGB values into the "safe range" for NTSC displays
* Can multiply alpha on load
* Can flip the image vertically
* Can compress and upload any image as DXT1 or DXT5
* Can convert the RGB to YCoCg color space
* Will automatically downsize a texture if it is larger than GL\_MAX\_TEXTURE\_SIZE
* Can directly upload DDS files. Note: directly uploading the compressed DDS image will disable the other options (no flipping, no pre-multiplying alpha, no rescaling, no creation of MIPmaps, no auto-downsizing)