Chapter 3
Computer Graphics Software Standards

• Interactive Computer Graphics Systems
Two ways in which users interact with graphics systems
1. Programming

Fig 3.1 An interactive graphics system

Fig 3.2 Example code

2. Interactive computer graphics packages

Fig 3.3 “Black box” approach to graphical output

Fig 3.4 Interactive modeling program

• Device Independence allows a graphics application program to run on hardware of various types.
• For examples, the same graphics packages run in many different platforms.
• Device independence technique is supported by the following standards — Standard Graphics Functions.

• Graphics Standards Systems
  CORE (1977)
  GKS (Graphics Kernel System, 1984-85, Figure 3.7)
  PHIGS (Programmer Hierarchical Graphics System, 1984, Figure 3.8)
  OpenGL (Open Graphics Library, trademark of Silicon Graphics, Inc.)
  DirectX Software Development Kit (SDK, in Windows 95)

• Graphics data store in GKS (Figure 3.7) or in PHIGS (Figure 3.8) are distinct.

• High-Level Language Standard
  ANSI (American National Standards Institute)
  ISO (International Standards Organization)
• Logical Workstations
  - an abstract graphics device that provides a logical interface (i.e. data) between the application program and the physical device.
  1. input only, with at least one logical input device and no output capability
  2. output only, having only one display area available and no input capability
  3. input/output, combining the characteristics of both the above

• Device Independence (Figure 3.9)
  WCS (World Coordinate System)
  UCS (User Coordinate System)

![](image)

**FIGURE 3.9** Device independence in the process of picture creation.

• Output Functions
  Primitives drawing

<table>
<thead>
<tr>
<th>PHIGS Primitive</th>
<th>Definition</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Polyline</td>
<td>A point sequence defining a set of connected lines.</td>
<td>![Example of Polyline]</td>
</tr>
<tr>
<td>Polymarker</td>
<td>Symbols placed at specific locations.</td>
<td>![Example of Polymarker]</td>
</tr>
<tr>
<td>Text</td>
<td>A character string placed at a given location.</td>
<td>![Example of Text]</td>
</tr>
<tr>
<td>Fill area</td>
<td>A polygonal interior area, without edges, filled in a variety of possible styles.</td>
<td>![Example of Fill area]</td>
</tr>
<tr>
<td>Fill area set</td>
<td>A complex, polygonal area, with or without edges, filled in a variety of possible styles.</td>
<td>![Example of Fill area set]</td>
</tr>
</tbody>
</table>

**FIGURE 3.10** PHIGS output primitives.

<table>
<thead>
<tr>
<th>PHIGS Polyl ine Attributes</th>
<th>Attribute</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Linetype</td>
<td>Selects a line style, such as: solid, dashed, center.</td>
</tr>
<tr>
<td></td>
<td>Linewidth</td>
<td>Selects a line thickness as a multiple of a workstation's nominal line width.</td>
</tr>
<tr>
<td></td>
<td>Polyline Color Index</td>
<td>Selects a line color through a workstation color table.</td>
</tr>
</tbody>
</table>

**FIGURE 3.11** Attributes to polyline primitive.
• **Input Functions**

Locator – indicates a position and/or orientation in world coordinates

Valuator – provides scalar values

Pick – selects a displayed entity, providing a pick status and path

Choice – selects from a number of possible choices and returns a non-negative number

String – provides a character string

Stroke – provides a sequence of positions in world coordinates

**Nontraditional devices**

![Figure 3.12 logical input classes](image)

**Table: Logical Device or Logical Function vs. Physical Devices**

<table>
<thead>
<tr>
<th>Logical Device or Logical Function</th>
<th>Physical Devices</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1) Locator</td>
<td>Tablet</td>
</tr>
<tr>
<td></td>
<td>Mouse</td>
</tr>
<tr>
<td></td>
<td>Trackball or crystal ball</td>
</tr>
<tr>
<td></td>
<td>Joystick</td>
</tr>
<tr>
<td></td>
<td>Joystick switch</td>
</tr>
<tr>
<td></td>
<td>Touch panel</td>
</tr>
<tr>
<td></td>
<td>Sonoic tablet 2D/3D</td>
</tr>
<tr>
<td></td>
<td>Noll box</td>
</tr>
<tr>
<td>(2) Valuator</td>
<td>Rotary potentiometer (DIAL)</td>
</tr>
<tr>
<td></td>
<td>Slide potentiometer</td>
</tr>
<tr>
<td></td>
<td>Control dials</td>
</tr>
<tr>
<td></td>
<td>Levers</td>
</tr>
<tr>
<td>(3) Pick</td>
<td>Light pen</td>
</tr>
<tr>
<td></td>
<td>Data tablet stylus</td>
</tr>
<tr>
<td>(4) Choice</td>
<td>A separate bank of buttons</td>
</tr>
<tr>
<td>(5) String</td>
<td>Alphanumeric keyboard</td>
</tr>
<tr>
<td>(6) Nontraditional devices</td>
<td>Speech recognizers</td>
</tr>
</tbody>
</table>

**Figure 3.13** Correlation between logical and physical input devices.

**FIGURE 3.14 Operating modes for logical input devices.**

**Segment Attributes**

*(modify segment as a whole)*

Transformation → Translation, rotation, scaling

Highlighting → Normal or highlighted

Visibility → Visible or invisible

Priority → Front or back (position with respect to viewer)

**FIGURE 3.15 Global segment changes.**
Differences between GKS and PHIGS

Data Store (Figures 3.7, 3.8)

GKS provides segment to collect those graphical primitives
PHIGS contains structure element to represent output primitives, attributes transformations, application-specific data

FIGURE 3.16 Comparison of graphics data structuring in GKS and PHIGS.

FIGURE 3.17 GKS and PHIGS handling of segmentation.

FIGURE 3.18 Difference in attribute binding between GKS and PHIGS.
• Graphical User Interface (GUI)
  X-Windows (menu or display windows, SDI/MDI etc.)

• How does OpenGL work?
  • Generic implementations
  The typical program calls many functions, some of which the programmer creates, some of which are provided by the operating system or the programming language’s runtime library. Windows applications create onscreen outputs usually named Windows API, by means of, the GDI (Graphics Device Interface). The GDI contains methods that allow one to write text in a window, draw simple 2D lines, and so on.

![Figure 3.19 OpenGL in a typical application program](image)

Usually, graphics card vendors provide hardware driver that GDI combined with. A software implementation of OpenGL responses the graphical requests from an application and constructs color images of the 3D models. It then supplies the images to GDI for showing on monitor.

Microsoft Windows NT 3.5/4.0, Windows 95 (release 2), Windows 2000, XP, all contains support for common generic implementations of OpenGL. (on MMX-enhanced CPUs)
OpenGL Hardware implementations (accelerated implementation)

![Diagram of OpenGL hardware implementations](image)

The OpenGL API calls are passed to a hardware driver. This driver does not pass its output to the Windows GDI for display; the driver interfaces directly with the graphics display hardware. Sometimes, part of the OpenGL functionality is still implemented in software as part of the driver, and other features and functionality can be passed directly to the hardware. (OpenGL pipeline)

The OpenGL pipeline
Definition: pipeline – a process that can take two or more distinct stages or steps

![Diagram of OpenGL pipeline](image)

OpenGL API functions response to an application calls, the commands are placed in a command buffer that fills with commands, vertex data, texture data, etc. When the buffer is flushed, either programmatically or by the driver's design, the commands and data are passed to the next stage in the pipeline. “Transform and Lighting” is a mathematically intensive stage where points used to describe an object's geometry are recalculated for the given object's location and orientation, and how brightly the color should be at each vertex.