Graphics Systems (1)

Contents

Graphics Systems

- Definition:
  - Interface between application software and graphics hardware system

- Fundamentals
  - Output primitives
  - Primitive aspects
  - Primitive attributes
  - Output model
  - Coordinate systems and clipping
  - Input primitives
  - Input model
  - Storage

Graphics Systems - Requirements

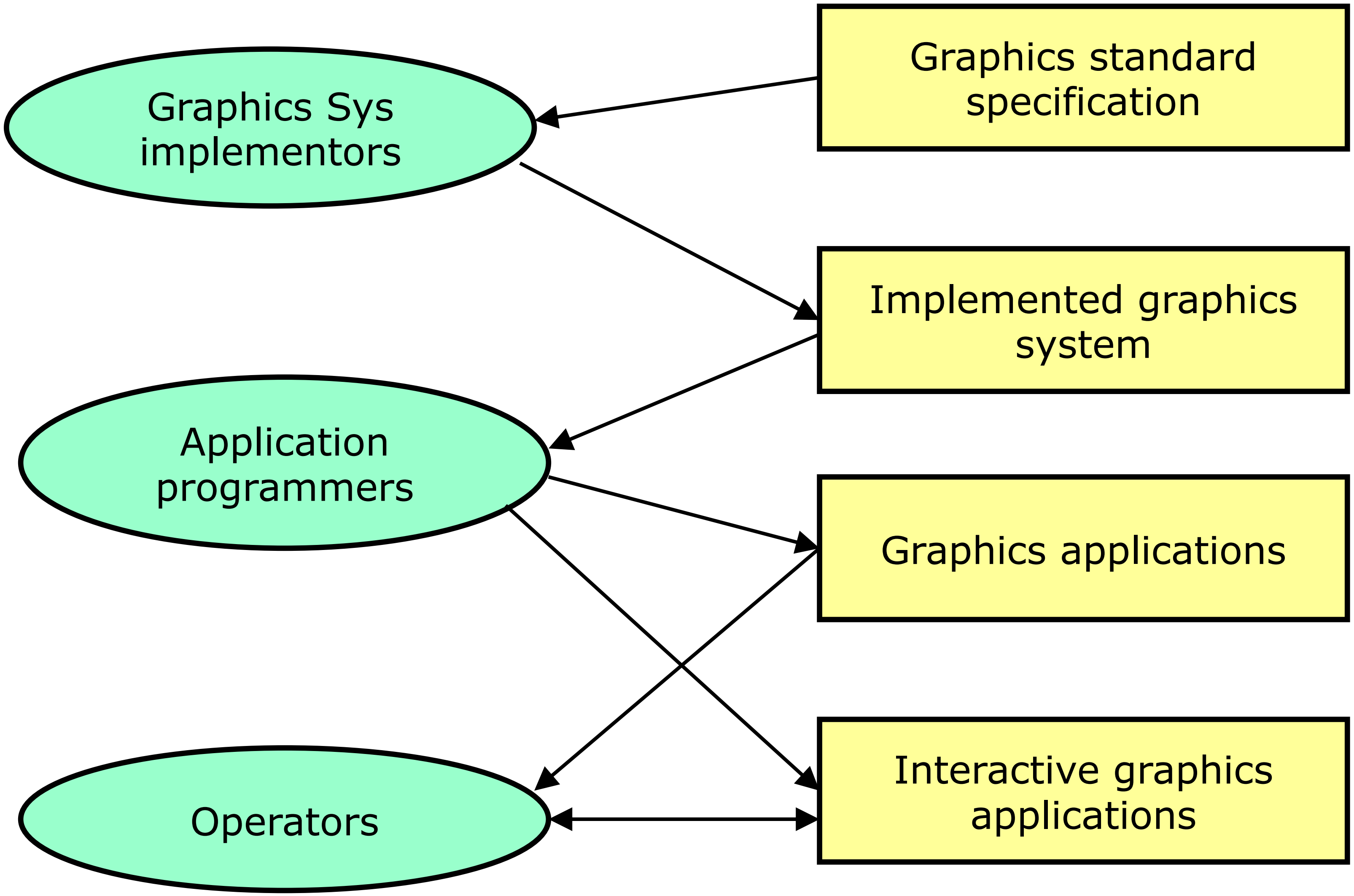
- Consist of input subroutines and output subroutines
- Accept input data and commands from user
- Convert internal representations into external pictures
- Graphics kernel assures
  - Application portability
    - Device independence
    - Language independence
    - Computer independence
    - Programmer independence
  - Application flexibility

Graphics Systems - Examples

- Core - 1977
- GKS (Graphical Kernel System) - 1985
- X-Window - 1986
- GKS-3D - 1988
- PHIGS (Programmers Hierarchical Interface Graphics System) - 1989
- CGI (device interface)
- CGM (metafile)
- CGRM (Computer Graphics Reference Model) - 1992
- IGES (Initial Graphics Exchange Specification) - 1980
- OpenGL (Open Graphics Library) - 1992
- DirectX

See CGI Historical Timeline, http://accad.osu.edu/~waynec/history/timeline.html

Roles in Graphics



CORE Standard

1. 3D Core Graphics System

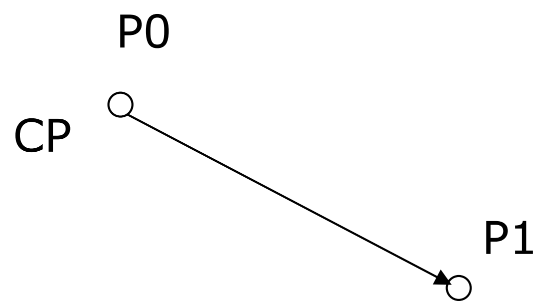
   1977, ACM SIGGRAPH (Special Interest Group on Graphics)

2. Six types of graphics output primitives:

   marker, polymarker, line, polyline, polygon, text

3. Current position (CP)

e.g.	move	x0,y0
	line	Δx1, Δy1



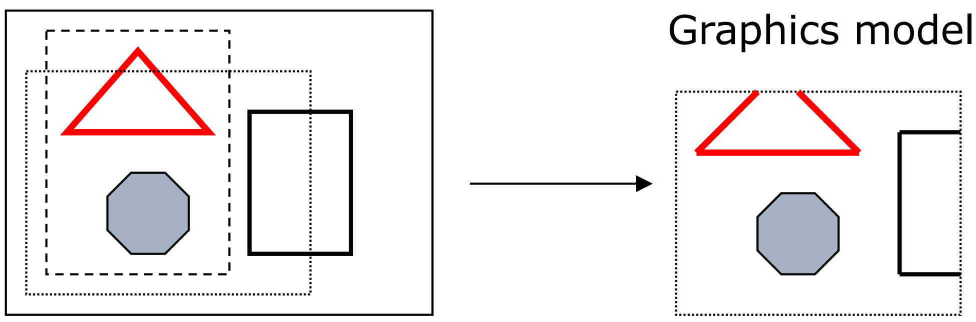
4. Two input modes:

   Sample

   Event

CORE Standard

4. Graphics model stores the clipped part of a graphical segment.



5. Space dimension

   CORE: 3D; GKS: 2D, 3D

6. Coordinate system

   Application program transforms NDC to WC

   GKS works on real world coordinate

7. Device control

   CORE identifies the output device

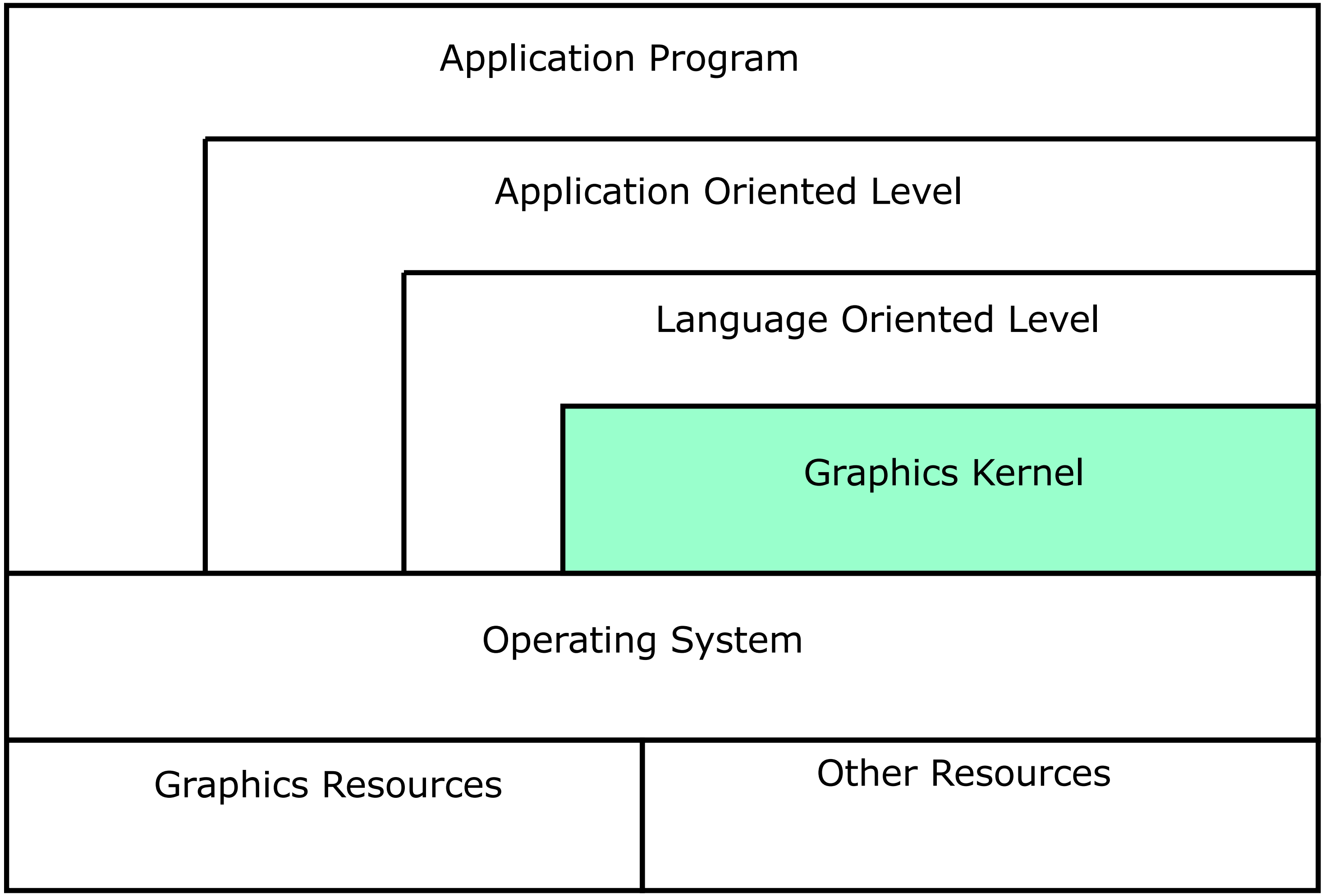
   GKS: workstations with associated list of attributes and window/viewport pair.

GKS Standard

- Graphical Kernel System
  - 1984 ('85) GKS 2D
  - 1988 GKS-3D

- Main concepts in GKS:
  - represents and generates graphics pictures
  - graphics presentation on particular workstations
  - control the workstations
  - control the inputs
  - graphics segments
  - metafiles

GKS – Functional levels



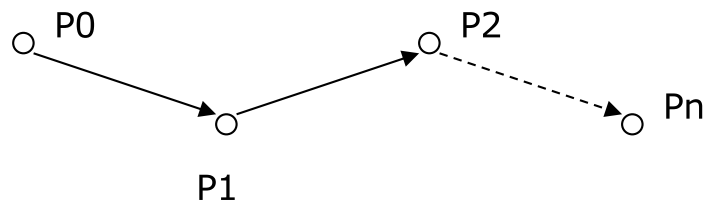
Graphics primitives in GKS

- Graphics primitive (type, attribute)
- Graphics attributes
  - local
  - global
- GKS primitives:
  1. Polyline
  2. Polymarker
  3. Text
  4. Fill-area
  5. Cel-array
  6. GDP (Generalized Drawing Primitive)

Graphics primitives in GKS

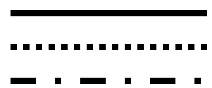
1. Polyline

   P0, P1, P2,…Pn

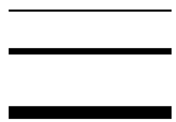


   Attributes:

   type of line: hidden, solid, dot, line-dot



   width:



   color: p x R + q x G + s x B

2. Polymarker

   Attributes:

   type: . + x * o

   dimension: * * *

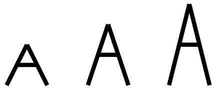
Graphics primitives in GKS

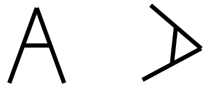
3. Text

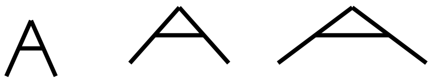
Attributes:

type: Roman Duplex, Roman Complex, *Italic*, Gothic

text precision: string, character, stroke TEXT

height: 

direction of characters: horizontal, vertical 

expansion 

direction of text: right, left, up, down

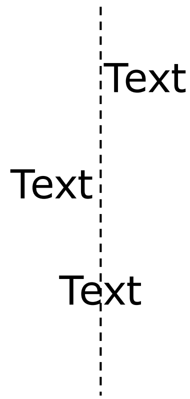
Text txeT

T
e
x
t

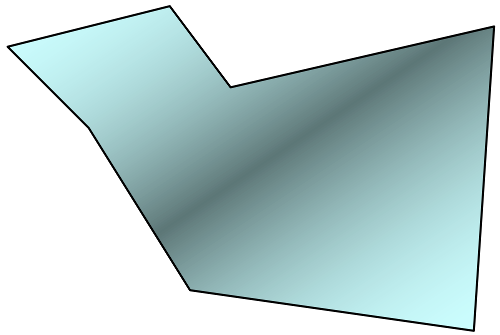
t
x
e
T

Graphics primitives in GKS

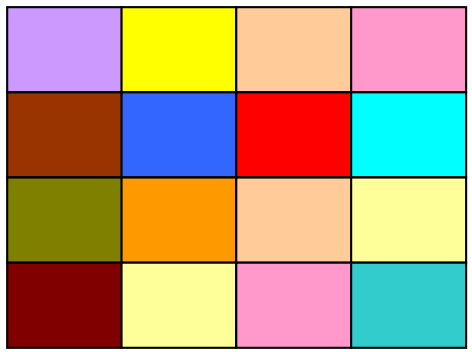
justify the text: left, right, center



4. Fill-area



5. Cell-array

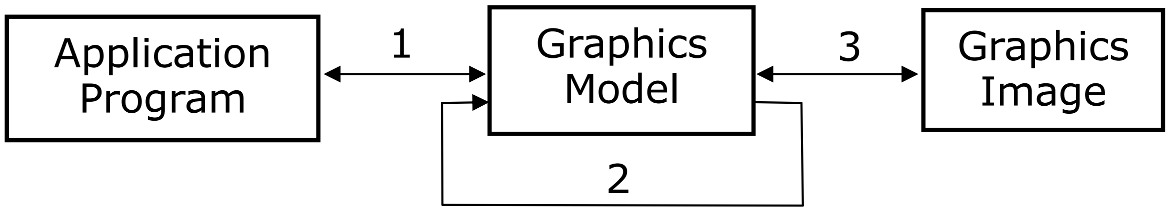


Graphics primitives in GKS

6. Generalized Drawing Primitives
   - Circle
   - Line
   - Interpolation Curves (Bezier, Hermite, B-Spline, etc).
   - …

Coordinate systems in GKS

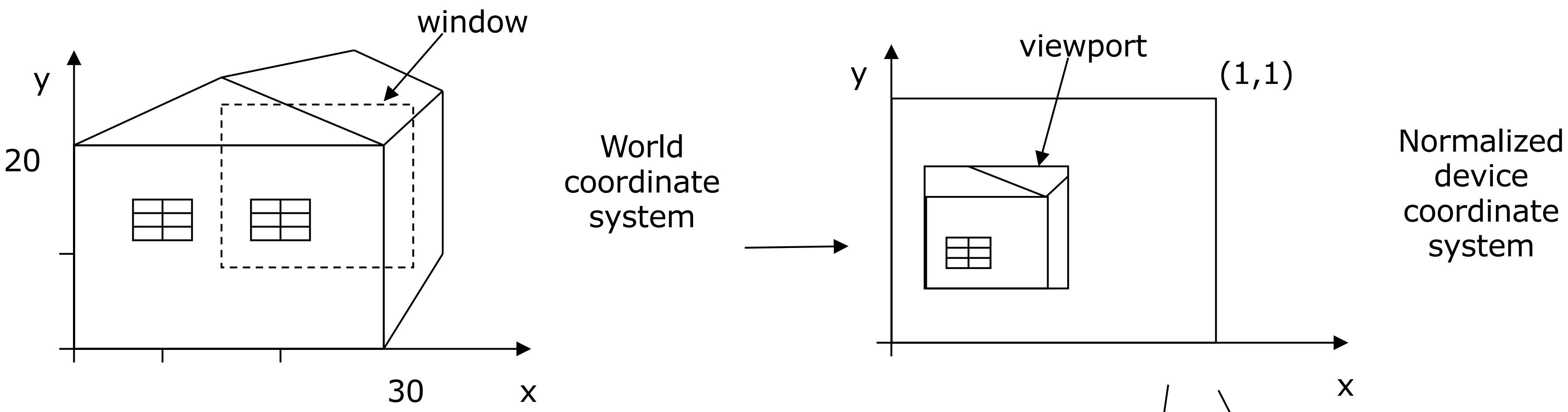
- World Coordinates (WC)
- Normalized Device Coordinates (NDC)
- Physical Device Coordinates (PDC)



World Coordinates	Normalized Device Coordinates	Physical Device Coordinates
Line 107.5,235.7 (m x m)	Line 0.35,0.71 [0,1] x [0,1]	

1. visualization and normalization transformations
   scales parts of a picture from window (WC)
   controls the placement into viewport (NDC)
2. segment transformations 2 x 3 matrix transformations
3. workstation transformations

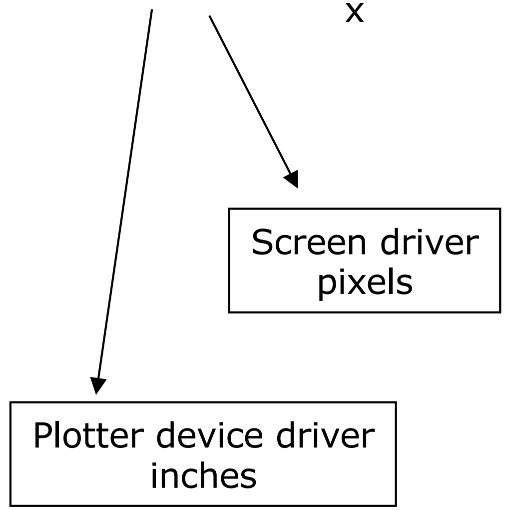
Transformation for a raster device



Window: rectangular area of interest defined by the world coordinates within the projection plane application space.
Viewport: rectangular area of interest within the normalized space (virtual and normalized screen).

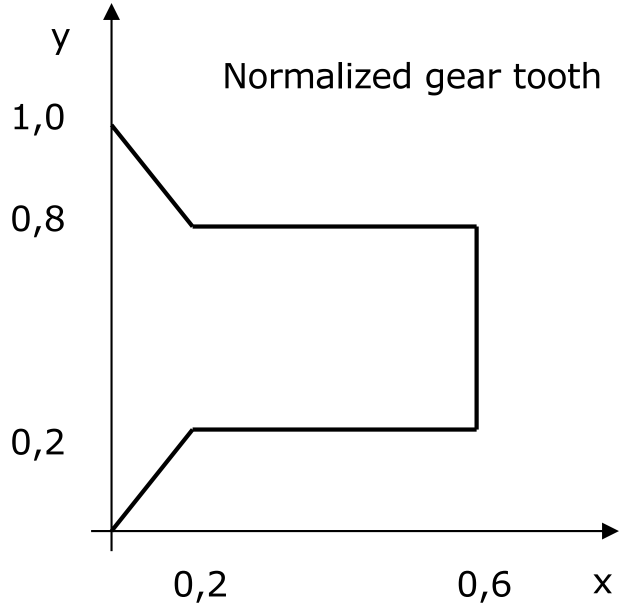
xPixel = (xResolution −1) *
(ViewportXmin + (ViewportXmax − ViewportXmin) *
(X − WindowXmin) / (WindowXmax − WindowXmin))

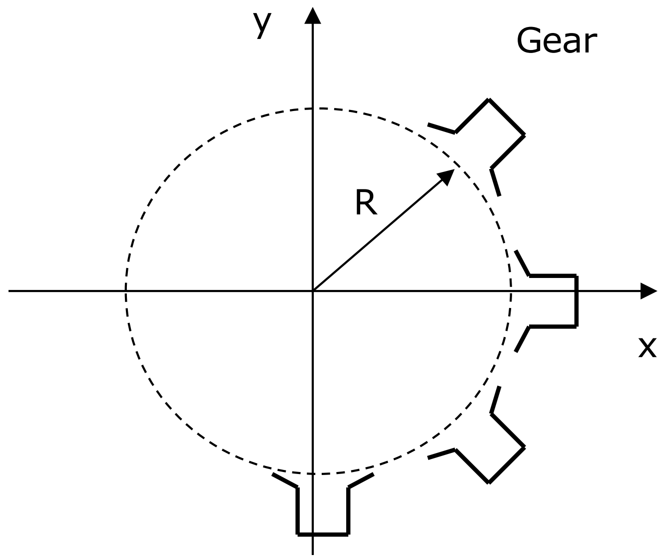
yPixel = (yResolution −1) *
(ViewportYmin + (ViewportYmax − ViewportYmin) *
(Y − WindowYmin) / (WindowYmax − WindowYmin))



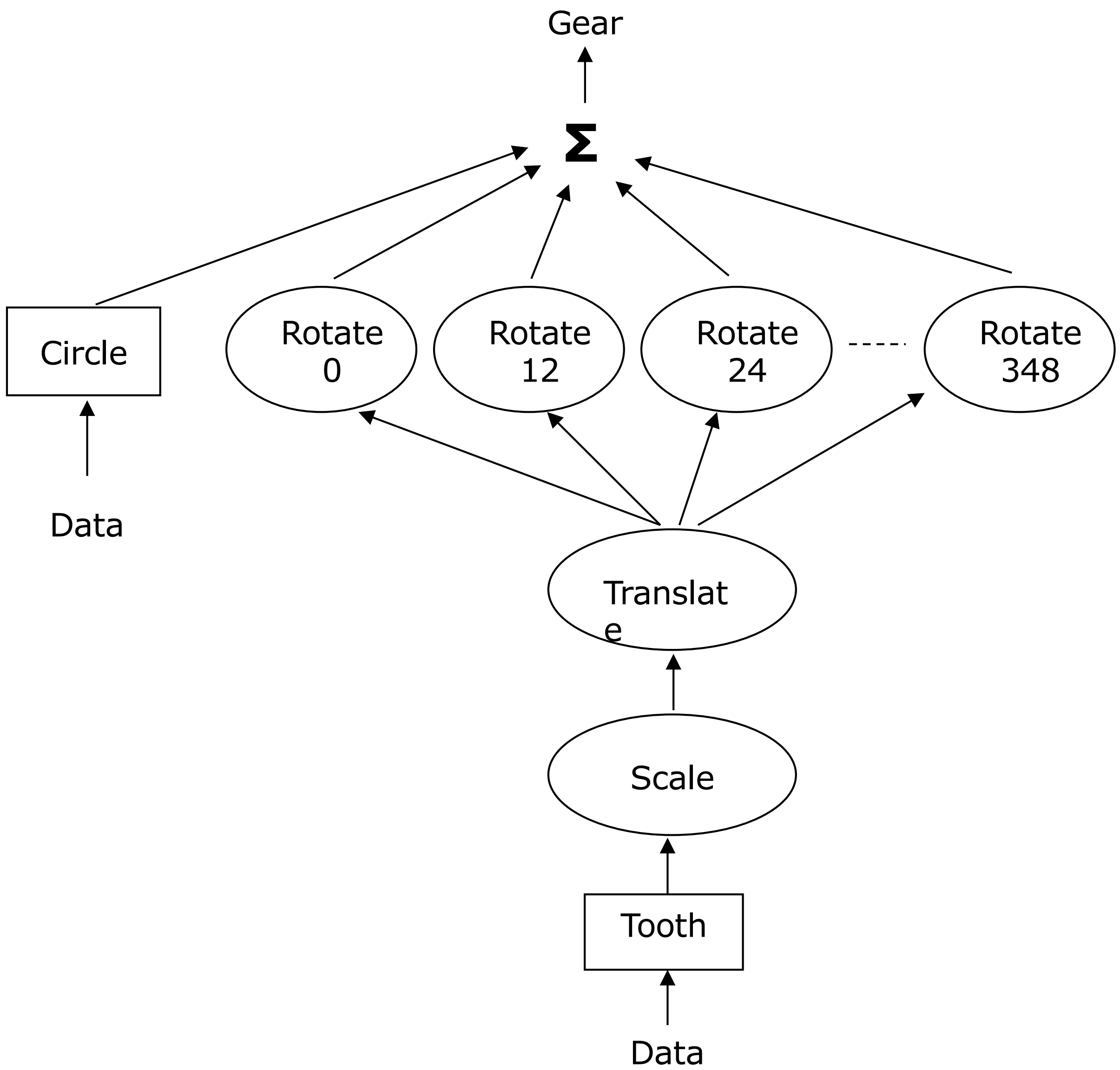
Graphics segments in GKS

- Group of primitives processed as a graphics unit
- Segment parameters:
  - tx, ty – translation parameters
  - sx, sy – scale parameters
  - a – orientation angle
  - visibility
  - priority
- Common attributes
- Common graphics transformations
- Graphics modeling
- Defined in NDC
- Instantiated in application program in WC





Graphics segments in GKS



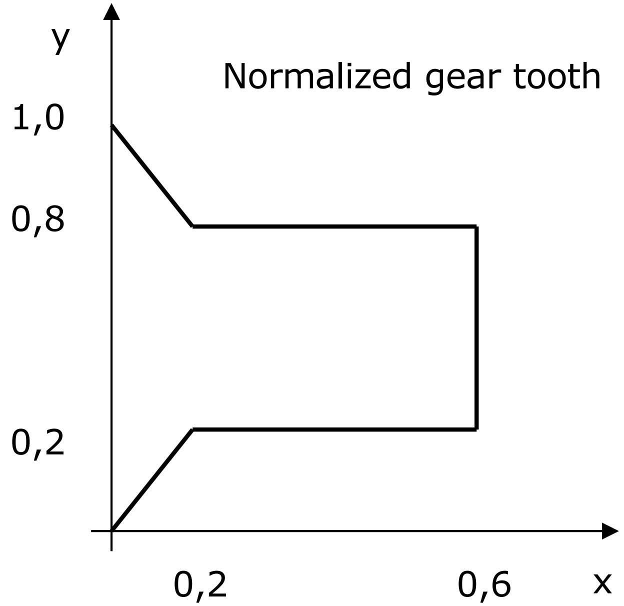
Graphics segments in GKS

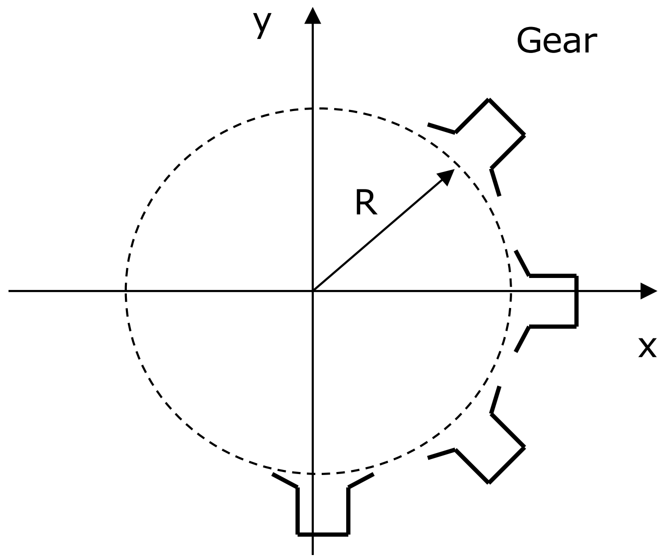
- Segment definition:

```
float   Xtooth[6], Ytooth[6];
/* initialize the Tooth structure */
OPEN_SEGMENT(Tooth)
        SET_POLYLINE_INDEX(Default);
        POLYLINE(6, Xtooth, Ytooth);
CLOSE_SEGMENT;
```

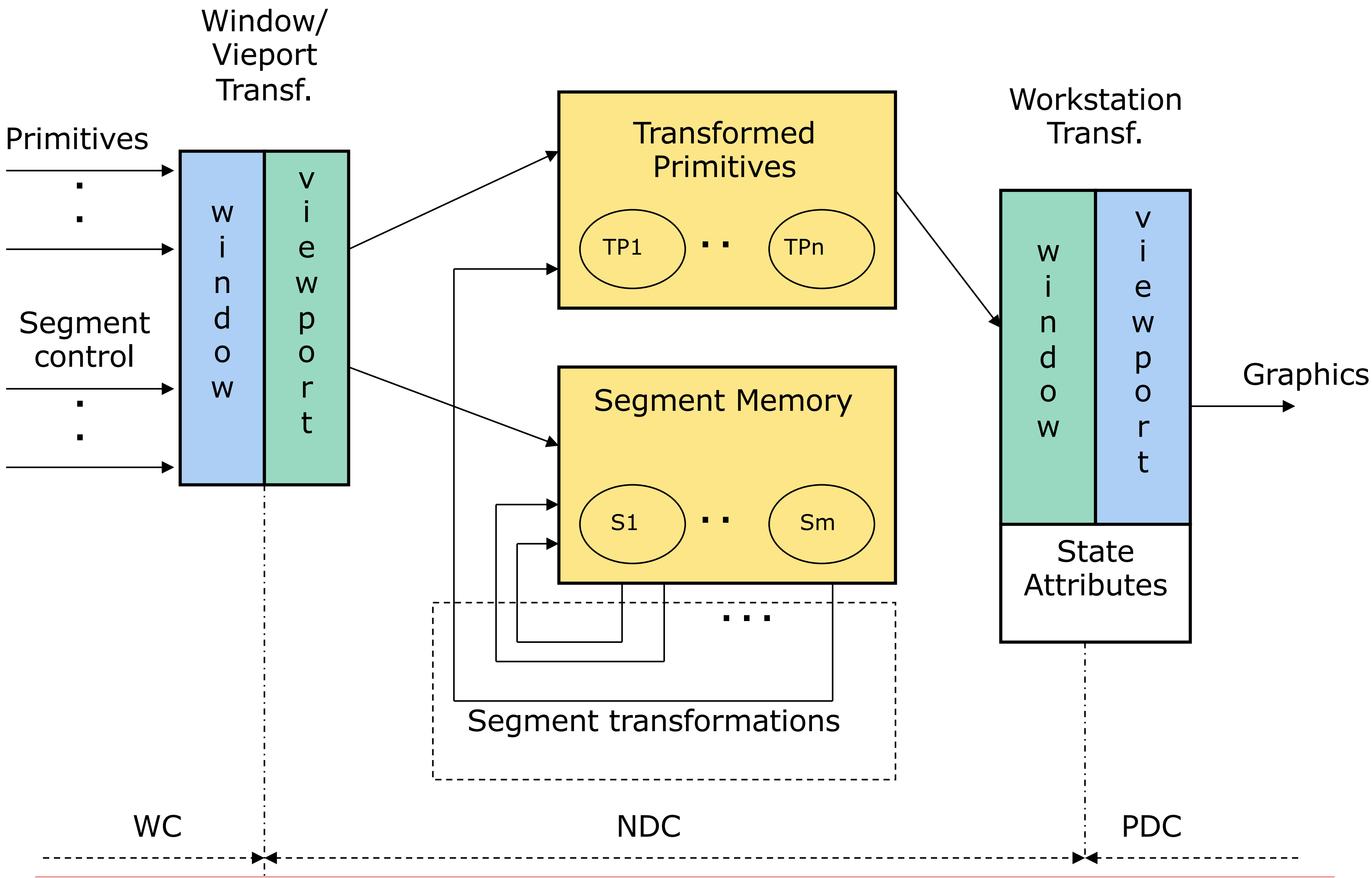
- Segment use:

```
. . .
SET_TRANSFORMATION_MATRIX(Matrix1, …
        Scale by R*COS(6), -R*SIN(6) …);
. . .
OPEN_SEGMENT(Gear)
for(j =1; j <30; j++){
        Angle = j * 12;
        INSERT_SEGMENT(Tooth, Matrix1);
        /* compute the complex transformation matrix */
        ACCUMULATE_TRANSFORMATION_MATRIX(Matrix1,
                … rotate by Angle…);
}
CLOSE_SEGMENT;
```





Graphics transformations in GKS



Graphics devices in GKS

- Output
- Input
- Input/Output
- WISS (Workstation Independent Segment Storage)
- MO (Metafile Output)
- MI (Metafile Input)

Graphical output devices

- Hardcopy device:
  - dot matrix printer
  - pen plotter
  - desk-top plotter
  - electrostatic plotter
  - laser printer
  - ink-jet printer
  - thermal transfer printer
- Display device:
  - monochrome and color CRT (Cathode Ray
  - direct-view storage tube (DVST)
  - liquid-crystal display (LCD)
  - plasma panel
  - electroluminescent display (ELD)

Input devices

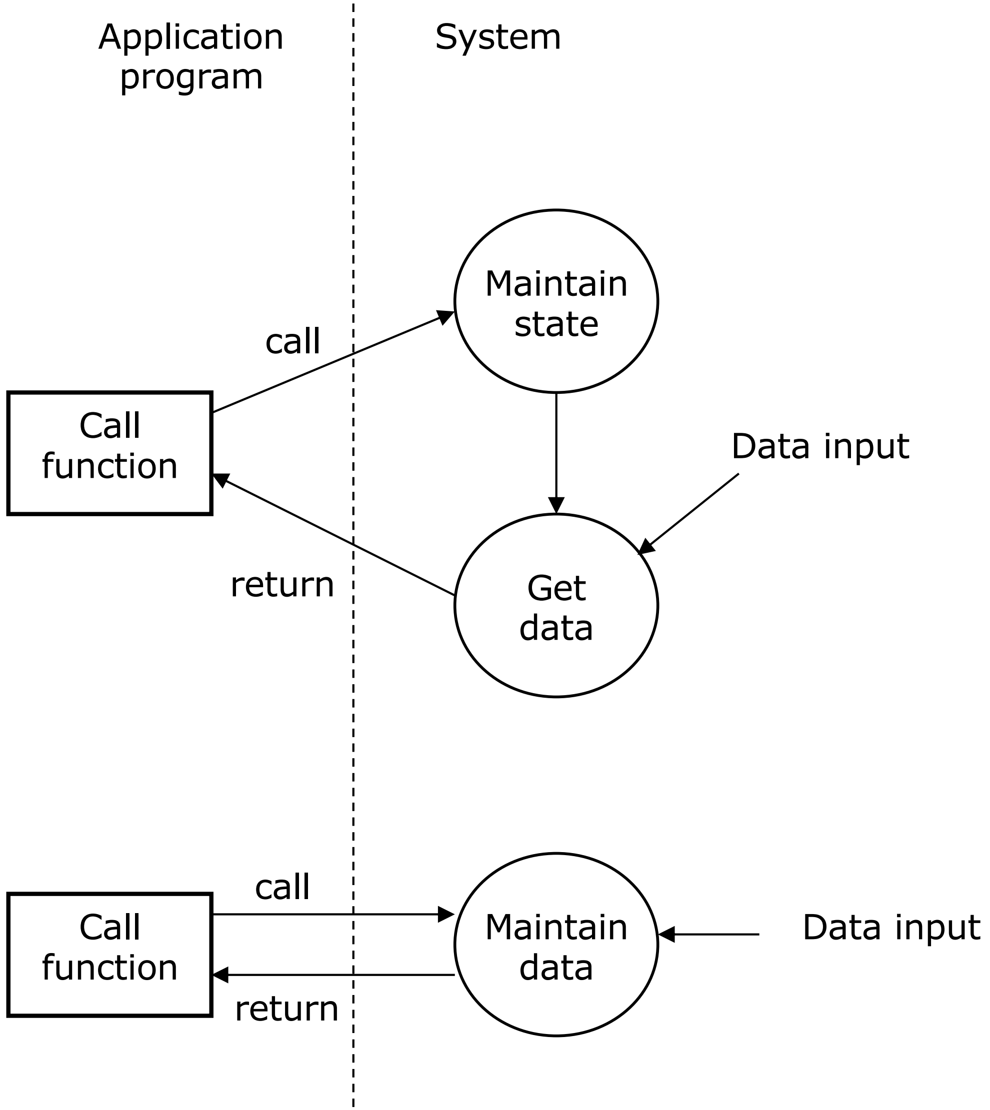
- Logical input device
  - Locator
  - Valuator
  - Choice
  - String
  - Pick
  - Stroke
- Physical devices:

locator:	graphic tablet, joystick
valuator:	analog to digital converter, potentiometer
choice:	keyboard, set of buttons (function keys)
pick:	light-pen
string:	keyboard
stroke:	mouse, track-ball

Input modes in GKS

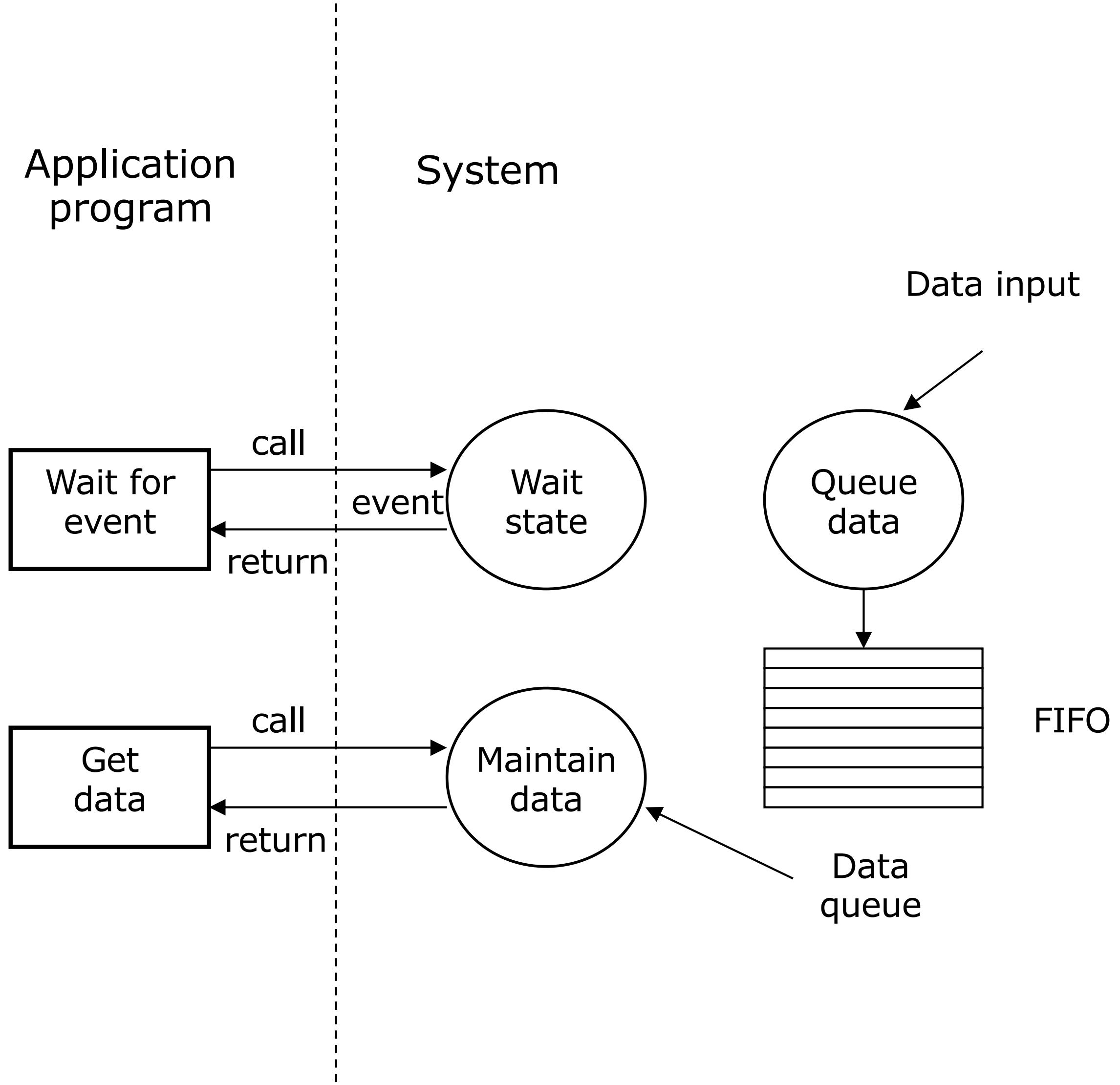
1. Request mode

2. Sample mode

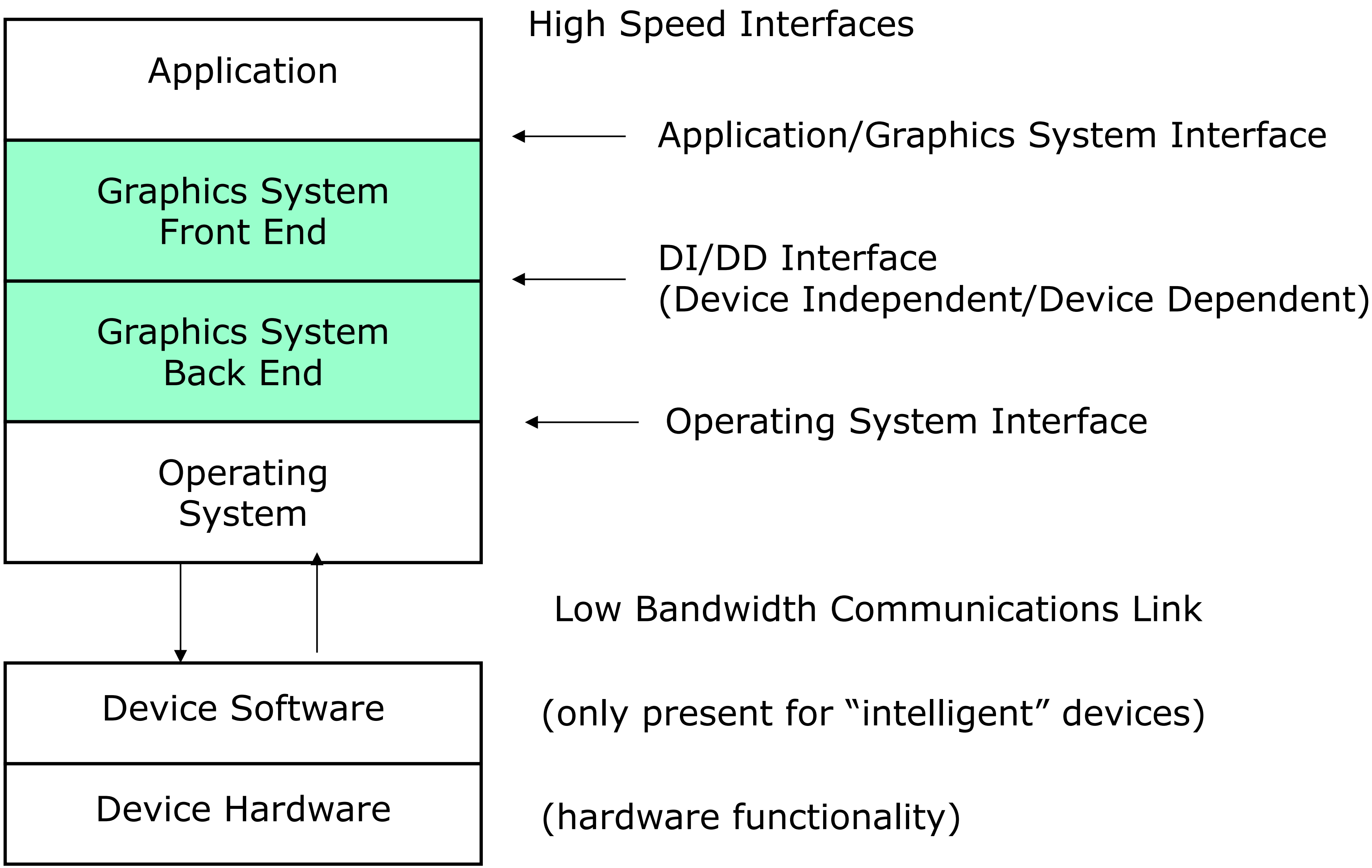


Input modes in GKS

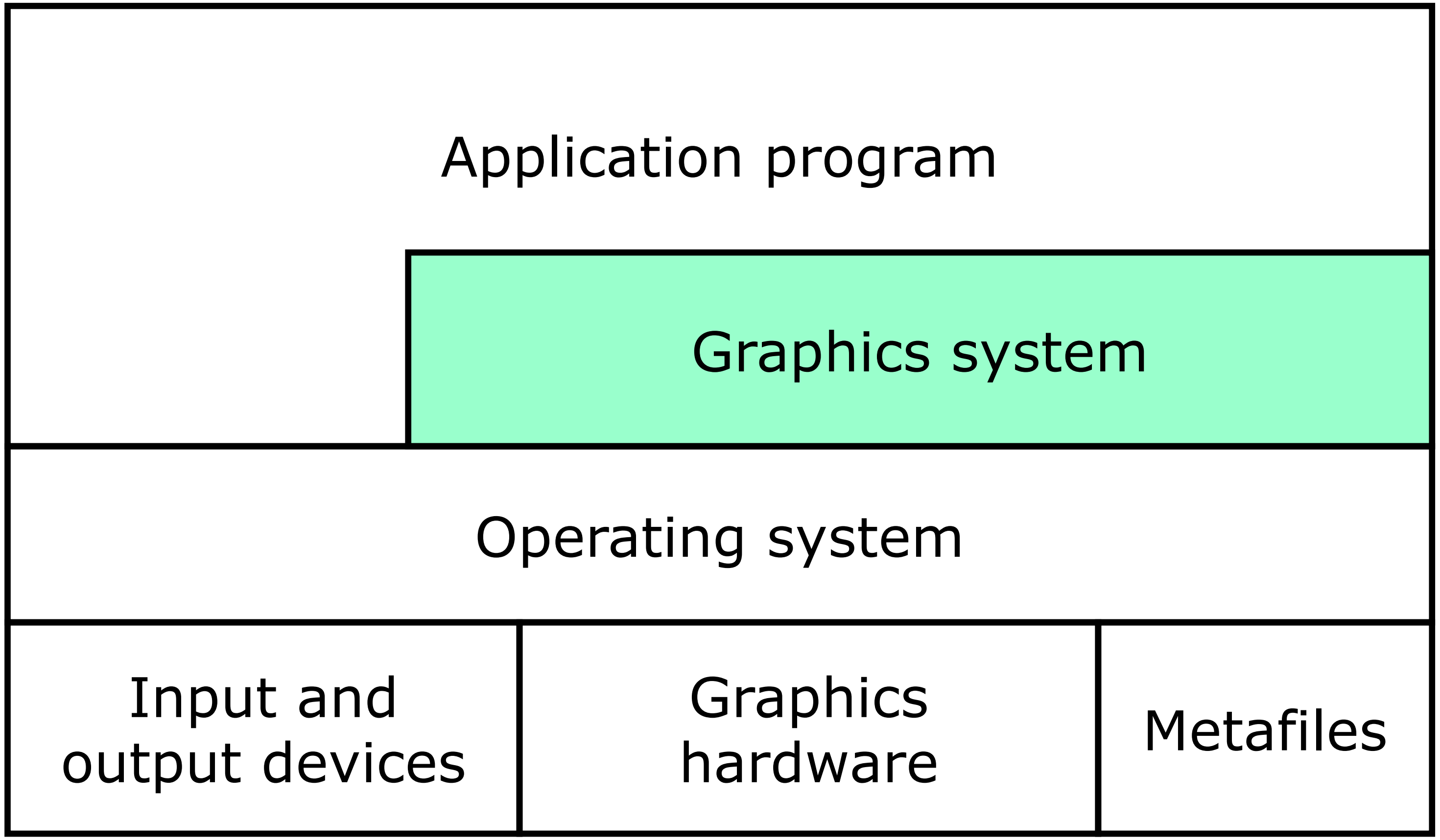
3. Event mode



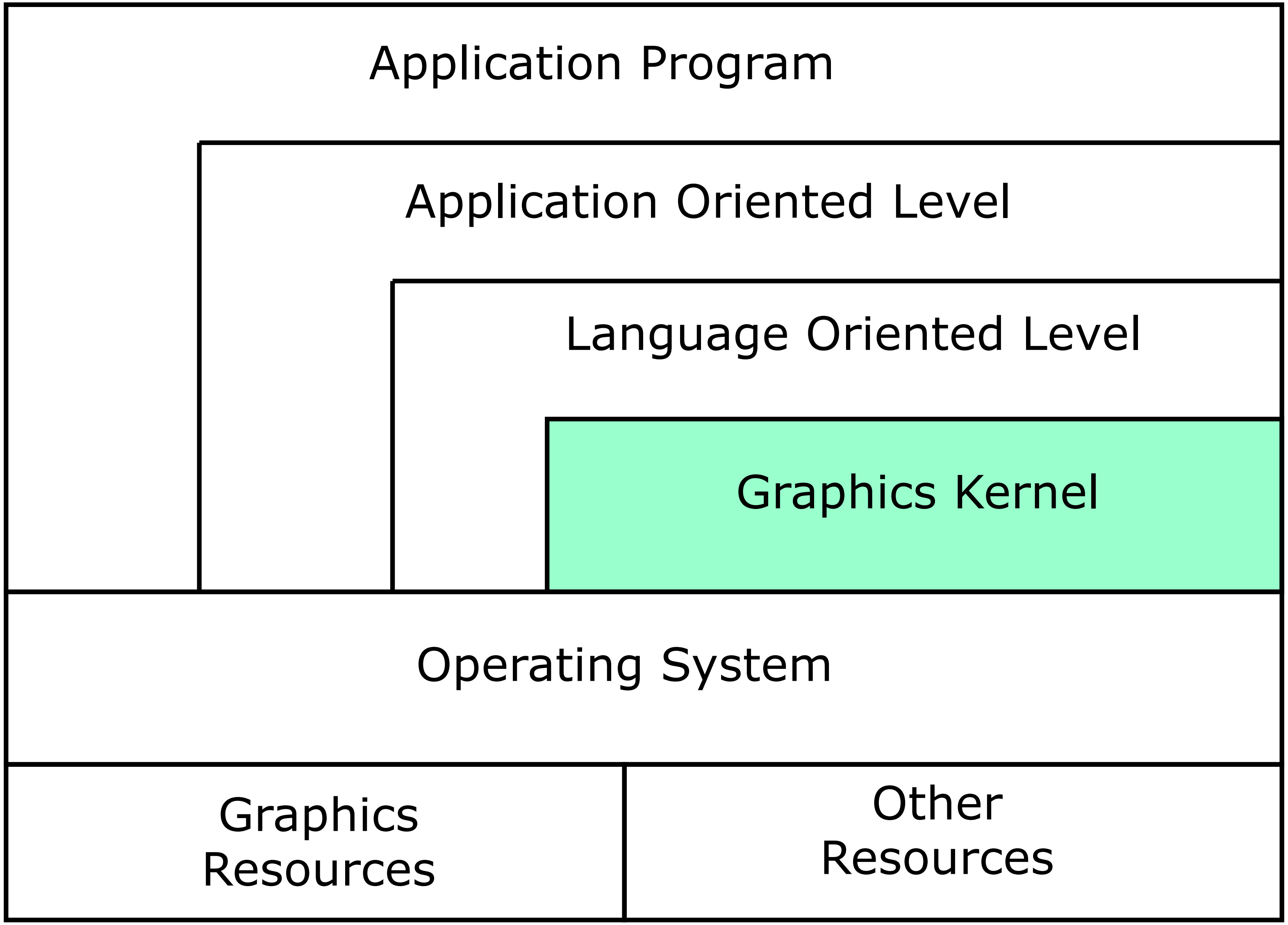
Layered functionality model



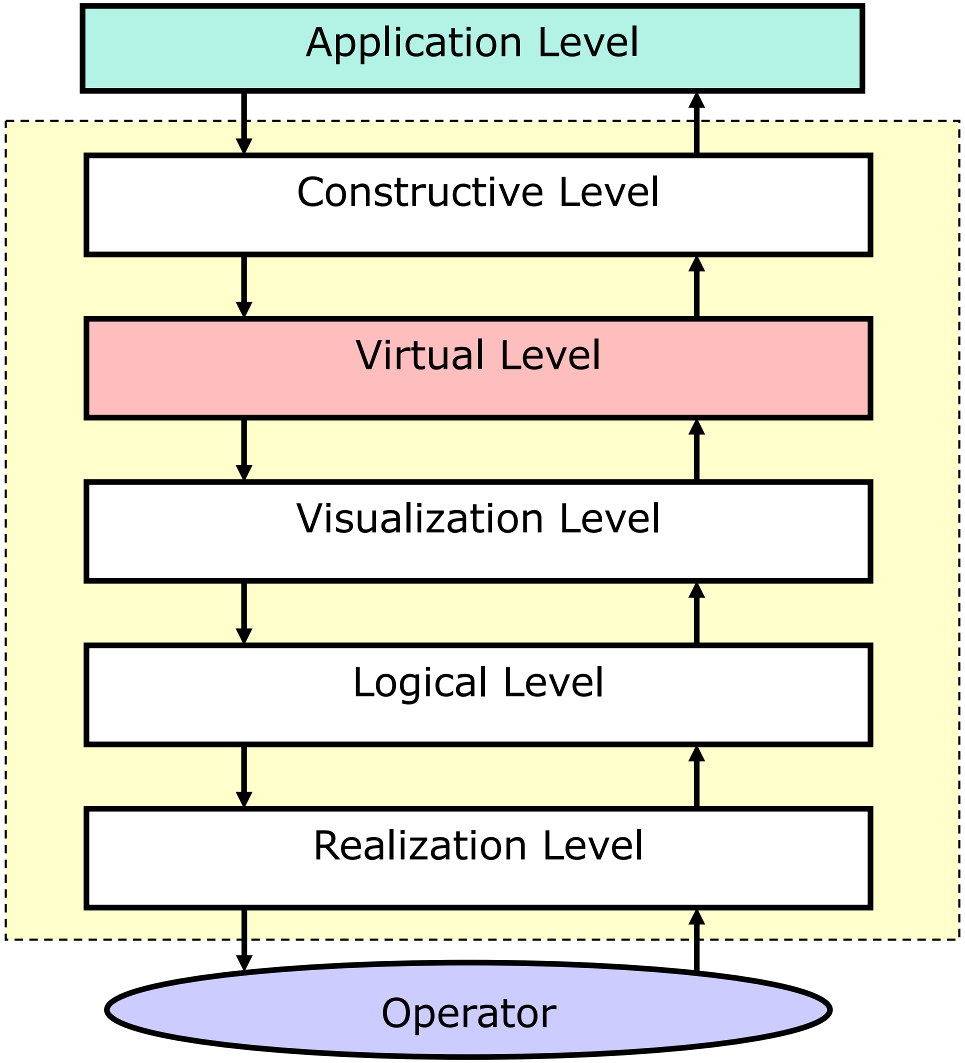
Interface of Graphics System



GKS Structure



Computer Graphics Reference Model



Block (1271.3, 56.2, 1472.45, 75.82)

Line (1271.3, 56.2, 1472.45, 56.2)
… Line (…)

Line (0.85, 0.15, 0.87, 0.15)
…

Line (0.85, 0.15, 0.75, 0.15)

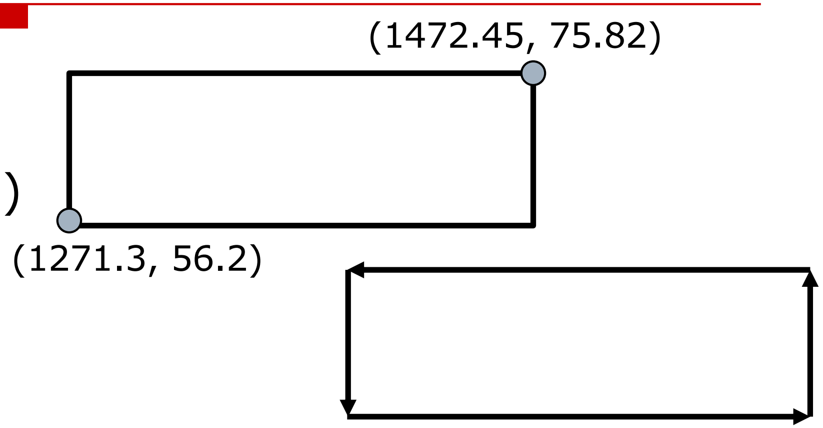
Move (0.85, 0.15), Line (0.75, 0.15)

Draw pixels, Get user inputs

CGRM levels example

1. Application Level

   Block (1271.3, 56.2, 1472.45, 75.82)



2. Constructive Level

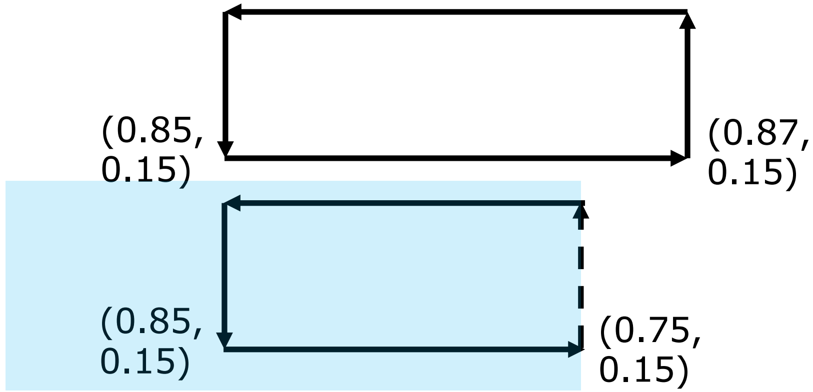
   Line (1271.3, 56.2, 1472.45, 56.2)

   … Line (…)

3. Virtual Level

   Line (0.85, 0.15, 0.87, 0.15)

   …

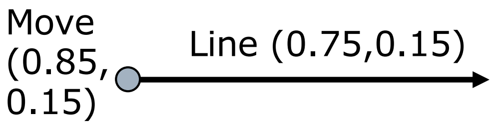


4. Visualization Level

   Line (0.85, 0.15, 0.75, 0.15)

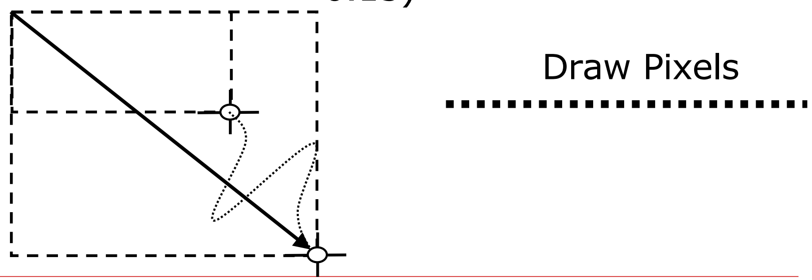
5. Logical Level

   Move (0.85, 0.15), Line (0.75, 0.15)



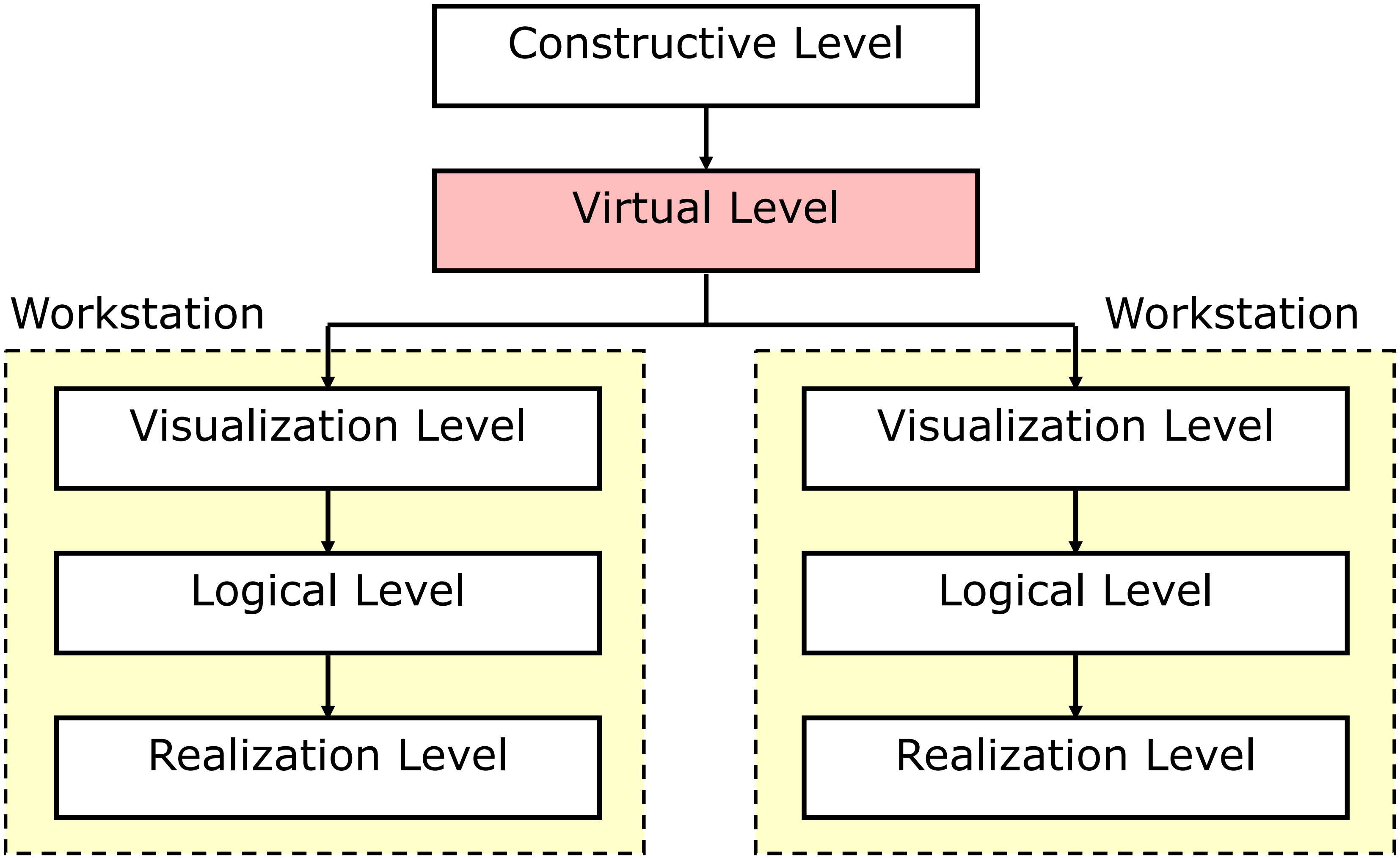
6. Realization Level

   Draw pixels, Get user inputs



7. Operator

CGRM on the GKS-3D workstations

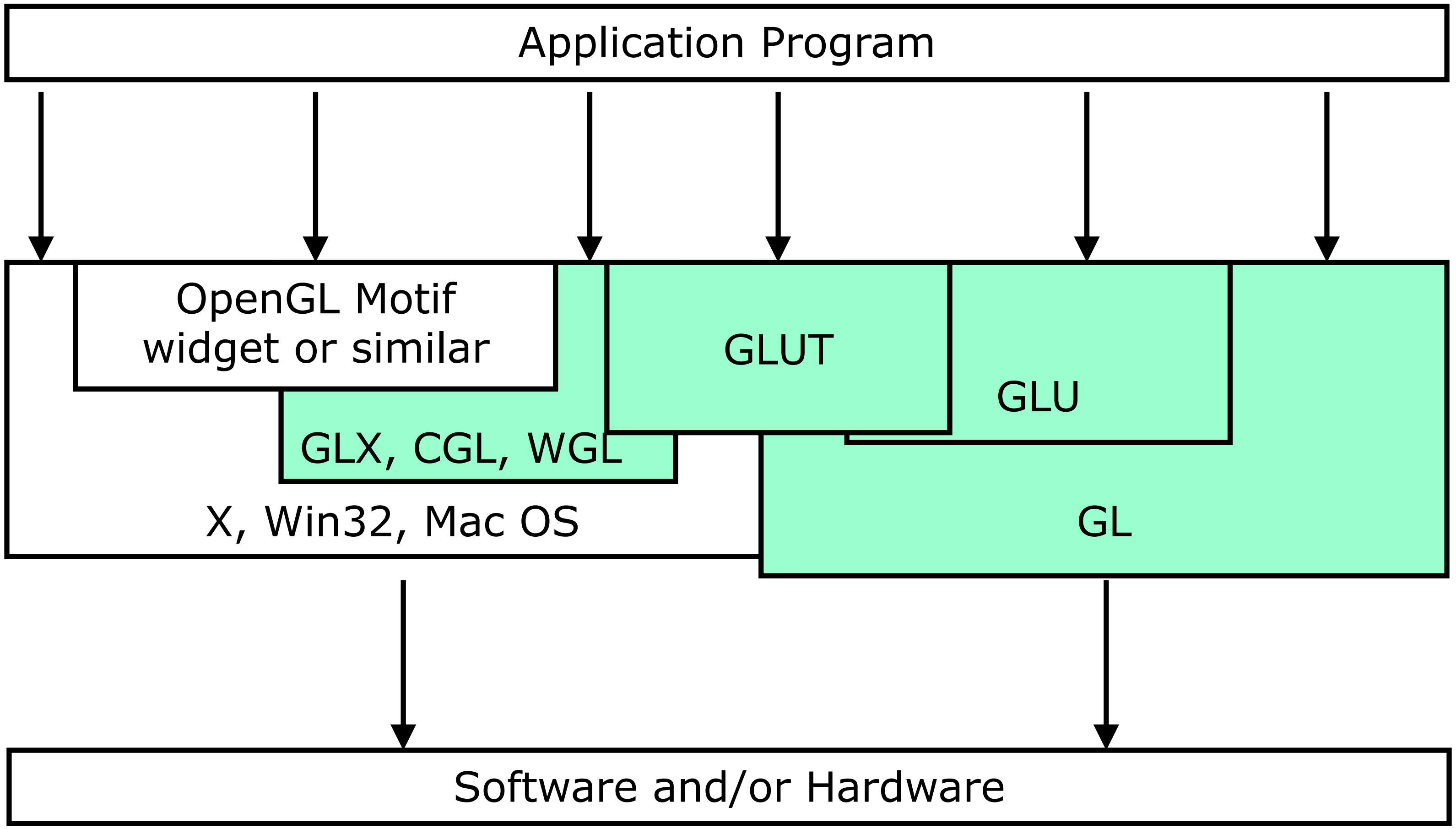


OpenGL

- API (Application Programmer Interface) to interactive rendering 2D and 3D vector graphics
- Silicon Graphics Inc. (SGI) started to develop OpenGL in 1991, and released it in 1992
- State machines
- Client-server model
- 250 distinct commands
- Object specification + image generation
- Simple primitives: points, lines, polygons

  (pixels, images, bitmaps)
- 3D rendering
- Reference: www.opengl.org/

OpenGL Levels



OpenGL- Related Libraries

- Window tasks
- User input
- Complex shapes

For portability, there are no commands for these

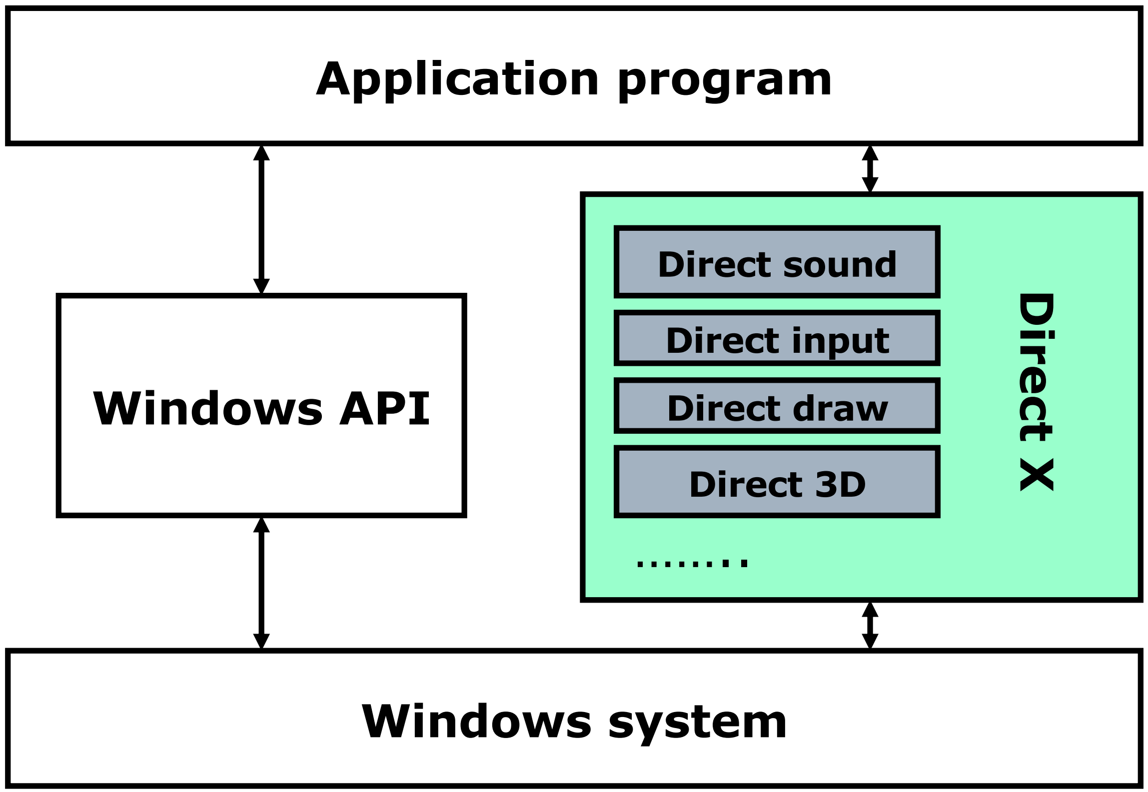
  - OpenGL Utility Library (GLU)
  - Window system support libraries: GLX / WGL / CGL
  - OpenGL Utility Toolkit (GLUT)
  - OpenGL User Interface (GLUI)
  - OpenInventor

OpenGL – language bindings

- More than 250 functions similar to those of the programming language C
- Language independent functions
- Language bindings:
  - JavaScript: WebGL (based on OpenGL ES 2.0 for 3D rendering within the web browsers). It uses HTML5 canvas element and is accessed through DOM (Document Object Model) interface.
  - C bindings:
    - WGL (Microsoft Windows interface to OpenGL),
    - GLX (X11 interface to OpenGL)
    - CGL (Mac OS X interface to OenGL)
  - C binding provided by iOS
  - Java and C bindings provided by Android

DirectX

- Direct control of graphics hardware (1995, Windows Games SDK)
- Direct control of keyboard, mouse, video cards, and sound devices



Questions and proposed problems

1. What are the main goals of a graphics standard. Exemplify by the CORE standard.
2. Explain the functional levels in the GKS Graphics System. Why the Application Program accesses the Kernel by other two levels? Why the Operating System separates the functional levels and the graphics resources?
3. What is the difference between the world coordinates, normalized device coordinates, and physical device coordinates? Why the graphics operations use normalized device coordinates?
4. Explain the differences between the concepts of window and viewport.
5. Explain the Request input mode in GKS. Exemplify a practical use case.
6. Explain the Sample input mode in GKS. Exemplify a practical use case.
7. Explain the Event input mode in GKS. Exemplify a practical use case.

Questions and proposed problems

8. Why the Application and the Graphics System are separate layers in the graphics system functionality model? Why this is a requirement?
9. Explain the layers of the CGRM (Computer Graphics Reference Model) and exemplify the transformations for drawing a triangle ABC.
10. What is the main role of the Virtual Level? What coordinate system is used to represent the scene of objects?
11. Why the Visualization, Logical, and Realization levels are different on various workstations? Give some examples.
12. Explain the differences between Application Oriented and Language Oriented levels, in the GKS Functional Structure. Why the Graphics Kernel is separated by the Language Oriented level?
13. Describe the main features of the OpenGL technology.
14. What are the main differences between OpenGL and DirectX.
15. Explain the reason for the direct control within the DirectX technology.