1996-11-10 Bo Nordin (bosse&cb.uu.se)

Lecture notes, computer graphics (F06) (Foley 7, GL, GKS)

Why standards?

In the beginning everybody wrote hist own "hardware-specific" code.

Work to standardize API (application programmer's interface) started 1975. The standard distinguishes between *modelling* and *viewing*.

Now: *portability* through device, langugage, and programmer independence.

device independence - normalized coordinates
computer independence - all new computers will support a standard. or so we hope.
language independence - definition of *functionality* in a lang. ind. form (pseudo) binding.
programmer independence - don't use non-standard extensions

What about *flexibility*? Hard to achieve together with portability

GKS, GKS-3D, PHIGS, PHIGS+ (official standards)	-- intelligent systems
GL, PostScript (unoffical strandards)	-- -"-
X (unofficial standard)	-- less intelligent (no store)
(Xt/Motif/OPENLOOK toolkits on top of X)	
UI builders	

X

Low-level protocol client <----------> server --> device drivers

PEX and GLX are extensions to the protocol (this means that the server must be extended)
2 kinds of request: those requiring replies and those which do not.

Explain xhost +, setenv DISPLAY, etc

Standards - historical overview

There's an evolution, standards come and go, new standards take the best from old ones. I will describe three evolutionary steps: CORE -> GKS -> PHIGS -> openGL
GKS: no 3D, no support for 3D viewing.

Standards exist for file storage (CGM): several images/file, non-sequential access, non-graphics data

For the device level (CGI): wants to standardize the graphics routines used by GKS, etc. and eventually move them to hardware.

GKS

Started eearly 70's - ready 1985
Subroutine library
Functions standardized, *syntax* may differ (diff. languages: FORTRAN, Pascal, Ada, C)

GKS closed <-> GKS open <-> WS open <-> WS active <-> Segment open
define "open"

Logical workstation: >= 1 viewing surfaces & any number of input devices (screen,printer..)
(user coord ->) WC -> NDC -> DC (can define several and use)
A workstation in GKS -> workarea (several can exist on one terminal)

Functions: workstation def / window handling / primitives and attributes / input / segment / segment storage WISS (databases) / metafiles

New attributes: cell array, generalized drawing primitive
Segment
Saves structures at a low level, cannot change the information.

segment: segment transformation (after normalization)
visibility
highlighing
priority
cannot be changed once they are closed.
usage: animation (bgr & fgr diff segments)

picture update: ASAP, BNIL, BNIG, ASTI.

Each workstation has a graphics data store
Aspect binding: 2 different kinds of aspects (global and ws - all aspects is exactly one type)
global (statically,modal) vs. workstation (groups, bundles, index, dynamic)
global, once set will always apply to that primitive
ws dep aspec can also be individual (ASP)

6 input modes: locator, pick, choice, valuator, string, stroke (req/sample/event)

Levels of implementation
a) no input
b) only REQUEST input
c) REQUEST, SAMPLE, and EVENT input

0) minimal output - all primitives available but only default attributes
at least one settable normalization transformation
1) Can specify the primitive index representations, more than one output workstation at a time, segments, multilple normalization transf.
2) workstation independent segment storage (WISS)

SPHIGS -> PHIGS (Programmer's Hierarchical Interactive Graphics System)

Chapter 7
Hierarchical model. (parts of object may change, be edited, like procedures in progr.):
master -> instance (modeling transformations).

1/ build, modfiy, maintain a model (CSS), often need an application model, too.
2/ SPHIGS traverses the model (CSS) for display

CSS (central structure storage)
A *structure* is a sequence of *elements* (primitives, attributes, structure, labels)
Structure manipulation: open/edit/close, delete, insert elements (primitives,attributes), delete elements, *post* for display through a viewing operation
as GKS segments only one can be open at a time

structure ID's are integers

Primitives: 3D (2D counterparts exist in PHIGS) (pointList)
places elements in the currently open structure (does not draw)
SPH_polyhedron () defined by vertices and facets (in PHIGS: set of fill area)
Attributes are typed (one for line, one for marker, etc.)

Posting structures:
postRoot(int struct, int index) -> posting a structure network,
view index (is a structure element in PHIGS)
"executes" the elements, attributes are modal
unpost -> only removes from the screen

Viewing: view orientation matrix using VRF/VPN,/VUP
view mapping matrix: view volume and viewport
setViewRepresentation(int, voMatrix, vmMatrix, viewport)

Modelling transformations:
scale, rotateX/Y/Z, translate
composeMatrix
setLocalTransformation(matrix, type) /* REPLACE,PREDCONCATENATE*/

Define in WC:

```
void House {
 polyhedron();
}

main() {
 openStruct( STREET );
 House()
 setLocalTrans( );
 House();
 closeStruct();
 postStruct();
}
```

Hierarhcy:

```
main() {
 openStruct( HOUSE );
 polyhedron();
 closeStruct();
 openStruct( STREET );
 executeStruct( HOUSE );
 setLocalTrans( );
 executeStruct( HOUSE ):
 closeStruct( );
 postStruct( STREET );
}
```

We may define a door on the house in its own system, then place with a local transf. somewhere on the house. The door will be transformed with the house (composing)
This means that the primitives of DOOR was transformed twice.

But SPHIGS composes these two matrices. Uses a stack of matrices.

Attributes:
Can be set in a structure - fixed
Inheritance rules: inherits the state, can modify without affecting ancestors.
bound dynamically at traversal time

The screen is constantly updated: can change rendering mode

Structure network editing:

Like line editing (DRAW a structure)
element pointer (set to after the last element when opened)
```
 setelementPointer( int )
 offsetElementPointer( int )
 label( int )
 moveElementPointerToLabel( int )
 deleteElement(); /* current */
 deleteElementInRange(int , int);
 deleteElementBetweenLabels( int , int );
```

Input: locator
pick: (modelled objects) hard in 3D (cf. AVS) full *path* returned (struct id ain't enough (can be displayed in several places.
which level? (path gives us freedom)
setPickIdentifier(int) - modally within a structure, not inherited

Limitations: neither necessary (application can update, pick correlate, etc.) nor sufficient (applications need their own own database with duplication of data and synchronization problems.
animation with a lot of change between frames does not workl

GL

Use the hand-out