SPHIGS

CSC 292:
Computer Graphics

Randal C. Nelson

Geometric Modeling

A *model* is an abstraction useful for understanding some process or structure of interest.

We will deal with models for which computer graphical representations are useful.

- Organizational Models: Hierarchies, taxonomies etc.
- Quantitative models: Physics, chemistry, biology etc.
- Geometric models: Bridges, buildings, molecules, geographic structures, etc.

Graphical representation of components may provide

- Spatial layout and shape:
- Connectivity of components: Networks, hierarchies, etc.
- Data values: Temperature, flow, text labels, etc.

Hierarchical Organization

- It is often useful to represent complex objects in terms of parts, subparts etc.
- Structure can be represented by a tree, with nodes as objects, and links as inclusion relations.
- Or as a *directed acyclic graph* (DAG), which allows pieces to be reused.

Advantages of hierarchical organization:

- Complex objects can be constructed in modular fashion from basic building blocks.
- Increased storage economy through re-use of primitives.
- Easy update propagation.

Parameter passing

- Objects invokes as building blocks must be positioned, and often resized and reoriented as well.
- Makes sense to allow transformations to be associated with sub-objects to allow more efficient re-use.
- Basic definition is sometimes called the *master*, and the transformed invocation the *instance*.
- Transform is (mathematically) exactly the same as the viewing transforms we have seen, but called *modeling transform* to distinguish it.
- Transforms can be passed down from parent to child, allowing relationships to be defined in convenient, local coordinate systems.

Subsystems of Application program

- Build, modify, and maintain the model by adding, deleting or modifying information in it.
- Traverse the model to extract display information.
- Traverse the model to extract information used in analysis of modeled system's performance.
- Perform display and interaction handling.
- Perform miscellaneous housekeeping tasks.

"80/20 rule": 80 percent of the program code deals with modeling, 20 percent with display. Some people say graphics is ALL modeling.)

SPHIGS

SPHIGS = Simple Programmers Hierarchical Interactive Graphics System.

- Allows 3D graphical objects be hierarchically defined, displayed, and manipulated.
- Simplified version of PHIGS.
- Example of a *retained mode* graphics system. This means it keeps a record of all primitives and relations.
- Contrast to an *immediate mode* system (such as SRGP), where changes are made directly to the screen, but the source of those changes is discarded.

Central Structure Storage (CSS)

- Special purpose database that PHIGS uses to store information on displayed objects (somewhat analogous to a vector display list).
- Stored *structures* consist of a sequence of elements corresponding to objects and attributes that is scanned in order to produce graphical output.
- Structures are generated by in-order procedure calls. They can also be edited (in a primitive fashion).
- CSS generally contains multiple structures.
- Facilitates fast display traversal
- May duplicate geometric info in user model.
- Provides automatic pick-correlation.
- Editing abilities enable easy dynamic effects.

Limitations of CSS

- May still need application data structure.
- Sometimes duplication of information is not worth it.
- Editing tools are rudimentary.
- Some version os PHIGS offer immediate-mode output facility.

Opening and closing structures

- Structures are opened with

  `SPH_openStructure(structure_ID);`

- Currently open structure is closed with

  `SPH_closeStructure();`

- Only one structure open at once.
- IDs are integers.

Simple Primitives

Primitives are added to structure by calling routines that generate. Examples are:

- Points (Three floating point values.)
- Polylines.
- Polymarkers.
- Fillareas (Like a polygon. Not defined if vertex set is not planer.)
- Text (very limited).

Polyhedra

- A polyhedron could be defined by specifying each face or *facet* as a fill area.
- Costs unnecessary duplication.
- Instead, we have a special polyhedron datatype, defined as a collection of facets, each facet is a list of vertex pointers, each vertex pointer is an offset into a vertex list.
- Routine call to define a polyhedron is

```
SPH_polyhedron
    vertex_count, facet_count,
    vertices, facets);
```

- vertices is an array of 3D points.
- facets is an integer array of indexes into the vertex array, each facet terminated by -1, e.g., [1,2,3,-1,2,3,4,5,-1 ...].

Attributes

- Values changed by routine explicit calls during structure building.
- Attributes stay in effect until changed
- Can be geometric transform or appearance.
- Bound to primitives during display traversal.
- Separate colors for background, fill areas, and edges.
- Widths are non-geometric, nominal units (not pixels).
- No patterns.

Posting for Display Traversal

- Structures are not displayed until *posted* from a view.
- A structure or *root* is posted by the call

```
SPH_post(structure_id, view_index);
```

- The view index is an integer index into a table of predefined views, which tell how to map the world onto the screen.
- Roots can be unposted with an analogous call.

Viewing

- Think of a "Synthetic camera".
- 3D Viewport in NPC coordinates, which are fix-mapped to screen.
- System maintains a table of *view representations*, each of specifies a view volume, a 3D viewport, and list of posted roots
- View table entries set by routine call to `SPH_setViewRepresentation` that takes view-orientation and view-mapping matrixes, and 3D viewport bounds.

- Two other routines `SPH_evaluateViewOrienationMatrix` and `SPH_evaluateViewMappingMatrix` allow these matrices to be constructed in terms of previously discussed primitives.
- `SPH_evaluateViewOrienationMatrix` takes VRP, VPN and VUP.
- `SPH_evaluateViewMappingMatrix` takes a projection type, the PRP (projection reference point), front and back clipping planes, and the 3D viewport specification.
- Note that 3D viewport must be respecified because matrices do not encode clipping information.

Other Viewing Stuff

Multiple views

- Structures can be posted to multiple view table entries.
- Might be tiled, transparent.
- Display priorities: View 0 is handled first, etc.

Interaction with window managers

- Biggest square in window is whole NPC space.
- PHIGS lets this mapping be changed, so window is only part of NPC space.

Modeling Transformations

- Might want to scale rotate translate an object.
- Call to routine `SPH_setLocalTransformation` causes everything in a structure after the call to be transformed by the supplied matrix.
- Modifies *local matrix*
- REPLACE, PRECONCATENATE, and POSTCONCATENATE modes.
- Routines such as `SPH_scale` and `SPH_rotateX` allow such matrices to be set up.

- Transformation is done at display traversal time.
- For efficiency, compose matrices before traversal.
- Use *template procedures* to avoid rewriting code for complex, reused objects.
- To get different instances, reset local matrix, and call procedure.

Hierarchical Structure Networks

- So far structures can contain output primitives, appearance attribute changes, and transformation changes.
- We can also execute a *substructure*.
- Done by calling `SPH_executeStructure(struct_id);` where `struct_id` is the identification number of the structure.
- Differs from using a template procedure because structure is not rebuilt.

- Facilitates use of building blocks.
- Multi-level hierarchies are possible and useful.
- Example of top-down design, bottom up implementation.
- Interactive modeling programs facilitate construction of data hierarchies.
- Lack of parameter passing in PHIGS limits power: e.g., can't control robot thumb positions independently.
- Structure hierarchy vs. template procedure hierarchy: Trade off efficiency against flexibility (as usual).

Inheritance of Attributes

- In traversal of hierarchy, attributes are *inherited* from parents (color, local transform, etc.)
- Attributes are bound at traversal time.
- Modified attributes are "popped" when traversal returns from child, i.e., changes in children do not effect parents.
- Effective composition of transform matrices can be used to great advantage.

Display Updating and rendering

SPHIGS updates the screen whenever one of the following occurs

- An entry in the view table is changed.
- A structure is closed (after having been opened and changed).
- A structure is deleted.
- A structure is posted.

Rendering Modes

There are four rendering modes, set by the procedure `SPH_setRenderingMode`:

- WIREFRAME: Just draws edges
- FLAT: Puts in uniform colors on faces, and does hidden surface removal. Keep edges on to see different faces of same color.
- LIT_FLAT: Shades faces uniformly depending on angle from light. Generally turn edges off.
- GOURAUD: Interpolates shading across faces. Used when you want to approximate curved objects with polyhedra. Generally turn edges off.

Editing for Effects

- Dynamic effects (simple animations), can sometimes be efficiently produced by *structure editing*.
- Minor changes - edit internal structure.
- Major changes - common to delete and respecify.
- e.g., move or rotate an object by changing the local matrix

Editor Commands

- Editing protocol recalls old line editor.
- Uses *element pointer* to position in structure.
- Routine calls move pointer, delete, insert structure elements at pointer etc.
- Elements can be labeled and searched for.
- Useful strategy: Organize structures with macros into blocks or semantic units.
- Pretty primitive.
- Possible to suppress automatic redisplay.

Interaction in SPHIGS

- Provides keyboard and locator items similar to SRGP.
- Locator returns NPC coordinates and index of highest priority view (can do z in PHIGS).
- Also provides *pick correlation* and *choice* capability.
- Choice supports menus. Read about it in the manual.

Pick Correlation

Pick correlation in hierarchical representation is more complex than in simple 2D world.

- What element of hierarchy should be returned?
- Structure ID is not enough, since structure may be component of many structures.
- How is proximity defined?
- What structure should be returned if primitives overlap on screen?
- One convention: Wireframes return first encountered object, shaded representations return object closest to user.

In SPHIGS

- In SPHIGS, pick correlation is done with call to `SPH_pickCorrelate`.
- Called with NPC point and view index (from locator).
- Returns maximum depth (pickLevel), and array of structure elements at various levels.
- Deepest is primitive, all others are structure executions.
- It is possible to define *pick IDs* (pick labels) during structure definition.

Other Stuff

- SPHIGS provides *highlighting* and *invisibility* operations.
- Also allows *name sets* for associating object that are not part of the same structure.
- To permit interchange of structures between implementations, *archive files* are defined, which allow the CSS to be written and read.
- To transfer only images, *computer graphics metafiles* (ANSI CGM standard) are available.

Implementation Issues

- Display consists of depth-first, recursive descent of posted structures.
- Process of going from CSS or other basic data structures to pixels is referred to as *rendering* and, and the sequence of stages as the *rendering pipeline*.
- Floating point operations, up to NPC parallelepiped representation is sometimes called *geometry processing* .
- Producing pixels after geometrical transforms is called *rasterization*.
- For wireframes, easy just do 2D line scan conversion.
- Shaded renderings involve three processes: visible surface determination, scan conversion, and shading.

Traversal

- Easy way: redraw everything. May be slow.
- Selective redraw is quite hard. Maybe not worth it on high performance workstations.
- One idea: Extent checking - keep track of extent boxes, and don't draw any structure whose extent does not intersect the viewport.
- Also make use of *elision* - the decision not to descend further in a structure.
  - *Pruning*: Don't draw objects outside of viewport.
  - *Culling*: Draw approximations (e.g. boxes) for very small structures.

Other Issues

- Use double buffering to get good image transitions.
- Pick correlation needs proximity checking.
  - Analytic approach: compute distance to lines, points, polygons etc.
  - Clipping approach: Use hardware clipper with a pick rectangle.

Procedure and Data Hierarchies

- Data hierarchy is good for dynamics, using simple building blocks, efficiency. Lacks general control-flow and flexibility.
- Pure procedure hierarchy provides flexible and powerful parameter passing. Can be inefficient, animation is harder in general, pick-correlation is tough.
- Structure hierarchy is towards the data end of scale.
- Template procedure is a mix providing some of both worlds.
- Object-oriented environments useful as performance improves - display procedures, attributes etc. can be attached to objects.