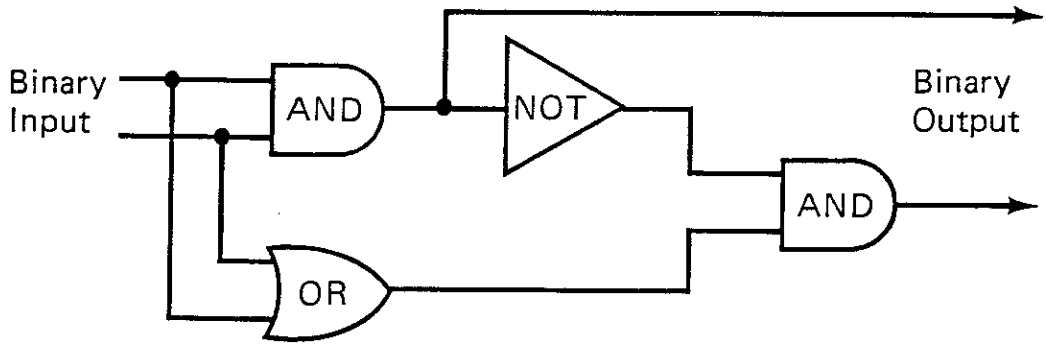
MODELING METHODS
(Chapter 15 in *Computer Graphics*)

- Basic Modeling Concepts
- Master Coordinates and Modeling Transformations
- Structured Display Files
- Symbol Operations
- Combining Modeling and Viewing Transformations

Basic Modeling Concepts

- modeling is the creation and manipulation of a system representation
- a model is any single representation
- the components of graphical models (also called geometric models) are represented with lines, polygons, etc.
- symbols are the building blocks from which models are built
- an instance is an occurrence of a symbol within a model



describing the model

- geometric information describing a model includes
  - coordinate positions
  - output primitives
  - attribute functions defining the structure of parts
  - data for constructing connections
- nongeometric data includes
  - text labels
  - algorithms describing the behavior of the model
  - rules for determining relationships between components

representing the model

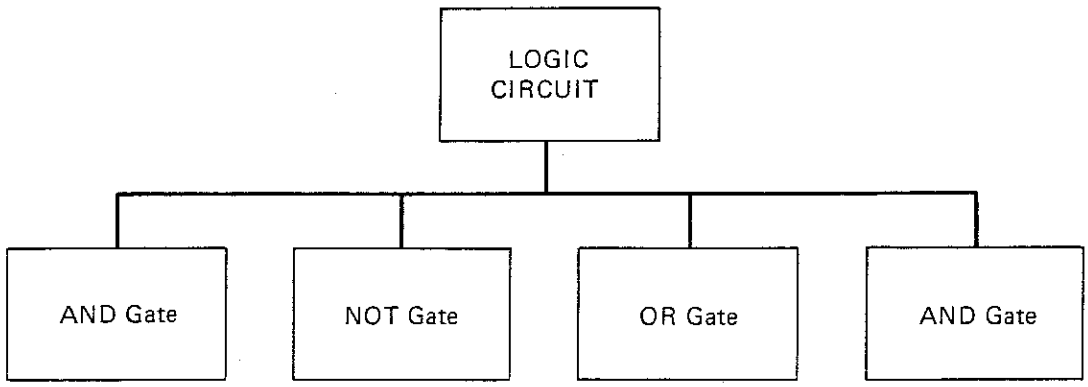
- in a table or other data structure

SYMBOL CODE	GEOMETRIC DESCRIPTION	IDENTIFYING LABEL
Gate 1	(Coordinates and other Parameters)	AND
Gate 2	⋮	OR
Gate 3	⋮	NOT
Gate 4	⋮	AND

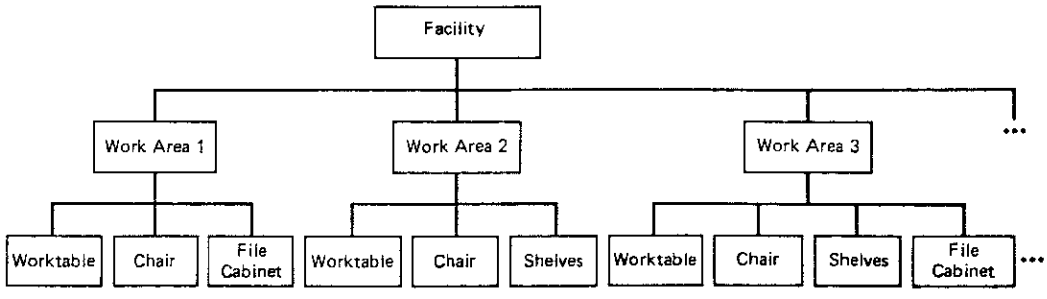
- in a procedure
- in both
- example (using the logic circuit)
  - geometric data to
    - position gates
    - draw gates
  - procedures to
    - draw connections
    - demonstrate behavior

symbol hierarchies

- flat hierarchy (the scene consists of four instances of symbols)



- a multi-level hierarchy made up of modules each of which is made up of other modules and instances of symbols (sometimes called groups and items)



modeling packages

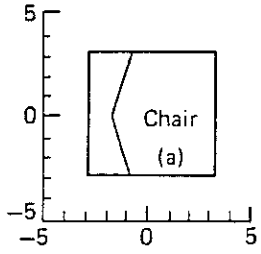
- modeling packages are separate routines to handle modeling procedures and data structures
- modeling packages and graphics packages can be interfaced
  - the graphics package generates and manipulates displayed information
  - the modeling package defines and arranges model representations
- modeling packages are often application specific
  - see figure 15-5 on page 313 for a sample output from a circuit design modeling package
  - see figure 15-6 on page 313 for a sample output from a molecular modeling package
  - see figure 15-7 on page 314 for a sample output from a plant design modeling package
  - see figure 15-8 on page 314 for a sample output from an office design modeling package

Master Coordinates and Modeling Transformations

- basic symbols are defined in an independent coordinate system called the master coordinate system
- consider two symbols defined in master coordinates

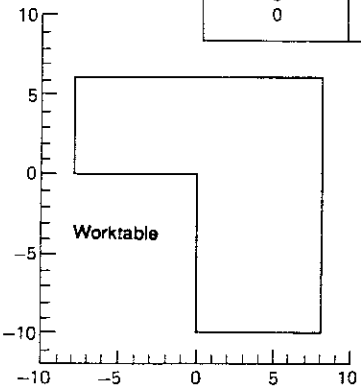
Arrays for Chair Coordinates

x_chair	y_chair
−3	−3
−3	3
3	3
3	−3
−1	−3
−2	0
−1	3

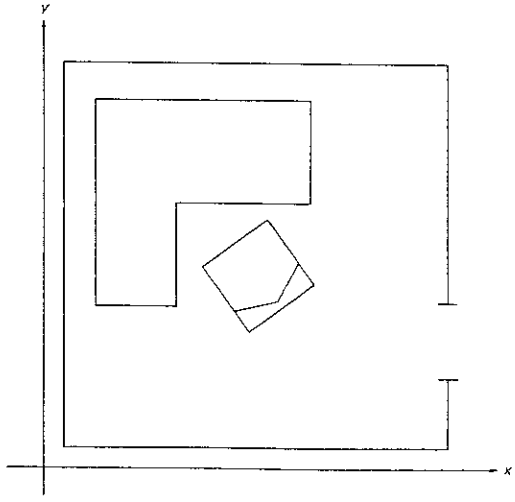


Arrays for Worktable Coordinates

x_worktable	y_worktable
0	0
−8	0
−8	6
8	6
8	−10
0	−10



- instances of these symbols occur in world coordinates



modeling transformations

- a symbol in master coordinates produces an instance in world coordinates when it undergoes a modeling transformation referred to as an instance transformation

$$(x_{\text{world}},\ y_{\text{world}},\ z_{\text{world}},\ 1) = (x_{\text{master}},\ y_{\text{master}},\ z_{\text{master}},\ 1) \cdot MT$$

- a modeling transformation may be selected by set_modeling_transformation (mt)
  - the transformation may be master-to-world or master-to-master
- a modeling transformation may be updated by
  - set_modeling_translation (tx, ty, tz)
  - set_modeling_scale (sx, sy, sz)
  - set_modeling_rotation (ax, ay, az)

modeling transformations, continued

- transformations are called in reverse order
  - if an object is to be rotated and then translated

    MT' := T·MT
    MT'' := R·MT'
    which is equivalent to
    MT'' := R·T·MT

modeling transformations, continued

- **solid modeling example**
  - **basic symbols, defined in procedures, include**
    - **cylinder**
    - **block**
    - **sphere, etc.**

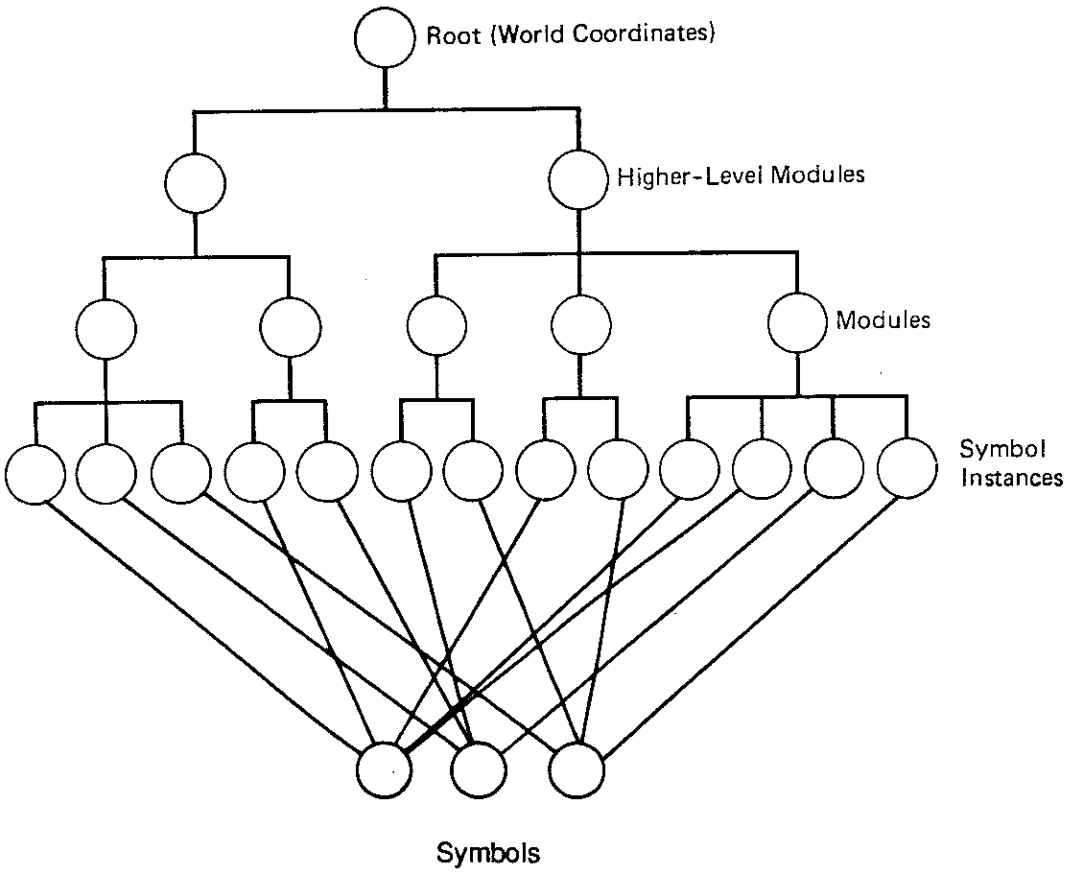
```
type
    instance = record
      symbol                          : integer;
      tx, ty, tz, sx, sy, sz, ax, ay, az  : real
    end;   {instance}
var
    instances : array [1..max_instances] of instance;

procedure display_instance;
    var k : integer
    begin
      for k := 1 to max_instances do begin
       create_segment (k)
       with instances [k] do begin
       set_modeling_transformation (identity);
       set_modeling_translation (tx, ty, tz);
       set_modeling_rotation (ax, ay, az);
       set_modeling_scale_factors (sx, sy, sz);
       case symbol of
            1 : cylinder;
            2 : block;
                .
                .
                .
         end  {case}
       end;  {with instances}
      close_segment
     end  {for k}
    end;  {display_instance}

procedure cylinder;
    begin  {definition of cylinder}  end;

procedure block;
    begin  {definition of block}  end;
```

modeling transformations, continued

- solid geometry operations may be available
  - union
  - intersection
  - difference
- the world coordinate description must be
  - transformed to viewing coordinates
  - clipped
  - mapped to a display device
- the modeling transformation and the viewing transformation can be combined

modeling symbol hierarchies

- a module is first defined as a list of symbol instances with transformation parameters
- this process continues up to the root of the tree, which represents everything in world coordinates

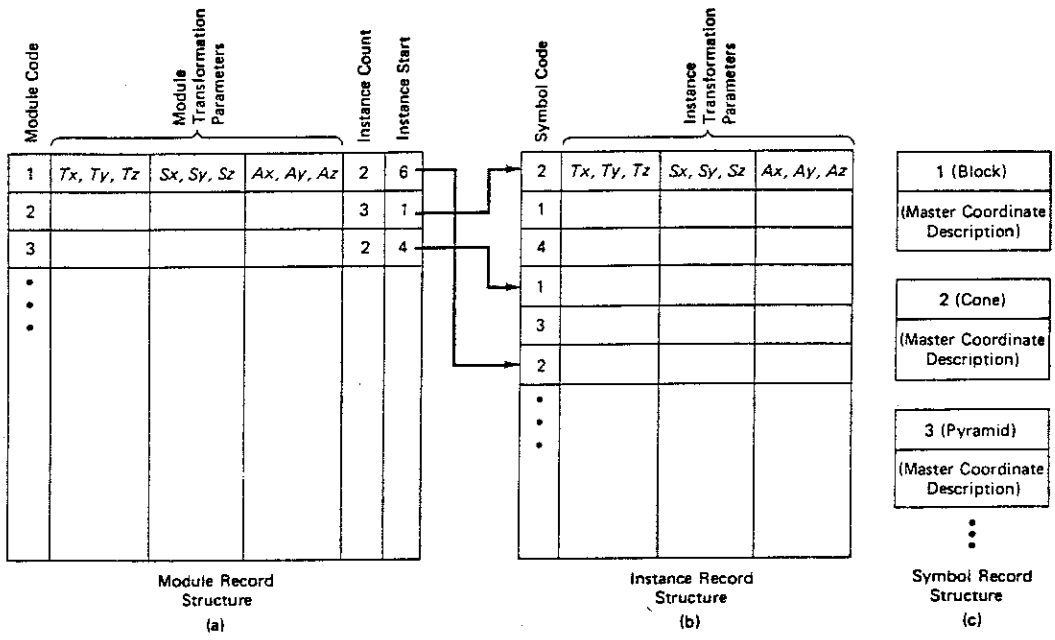


- each level can be thought of as a level of master coordinates

saving and restoring transformations

- a transformation higher in the tree is concatenated once with each transformation in its immediate subtree
- recalculation of higher transformations is avoided by
  - traversing the hierarchy in preorder
  - save_modeling_transformation (mt, m_stack)
  - restore_modeling_transformation (mt, m_stack)
- steps
  - save the current modeling transformation matrix
  - combine the instance transformations with the current modeling transformation
  - call the symbol procedure
  - restore the original modeling transformation

storing modules, instances and symbols



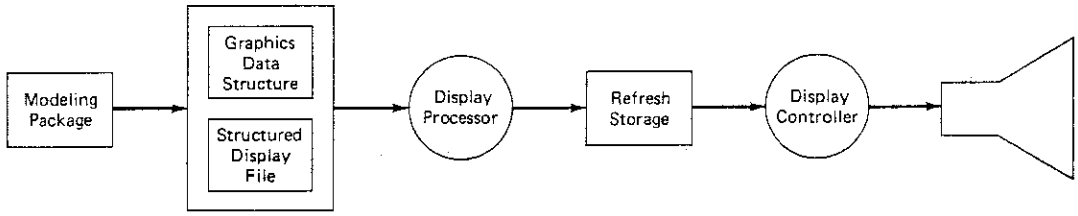
- alternative: store records in a tree structure

display procedures

- a display procedure call specifies a symbol name and an instance transformation
  display (symbol_name, sx, sy, sz, ax, ay, az, tx, ty, tz)
  - a convenient shorthand for creating symbol instances
  - flexibility in the order of transformations is sacrificed
  - save and restore are included implicitly

Structured Display Files

- a structured display file
  - reflects symbol and module relationships
  - is accessed by the display processor to create and update display information in the refresh storage area



  - a graphics data structure may contain non-geometric data
  - a structured display file is designed to support rapid changes

Symbol Operations

- symbols and modules can be defined with segment-type operations
  - create_symbol (id);
    .
    .
    .
    close_symbol;
- symbols can be included in segments
  - **create_module (12);**
    .
    .
    .
  - save_modeling_transformation (mt);
    .
    {perform instance transformations}
    .
  - insert_symbol (5);
  - restore_modeling_transformation (mt);
    .
    .
  - **close_module (12);**

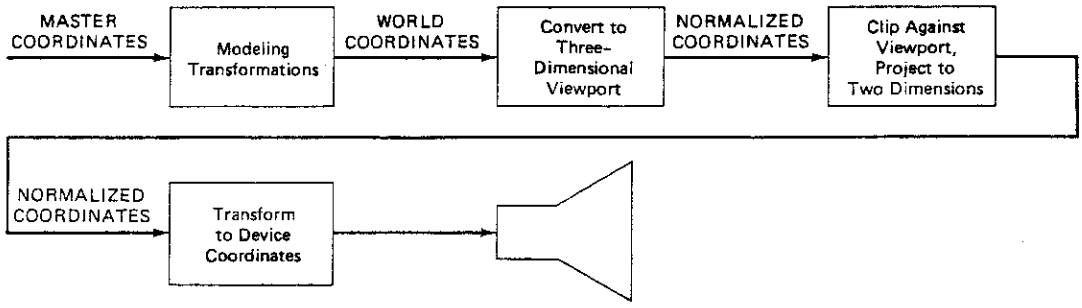
Symbol Operations, continued

- **hierarchic modules can be defined**
  - create_module (9);
    .
    .
    .
  - insert_module (7);
    .
    .
    .
  - insert_symbol (11);
    .
    .
    .
  - close_module;

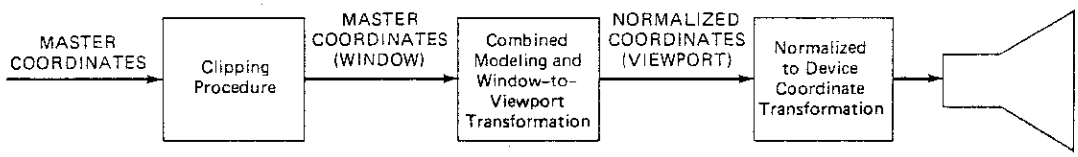
advantages of symbols and modules over display procedures

- permanent symbol libraries can be created and stored
- transformation parameters can be included with the insert operation
- interactive picture manipulation is supported

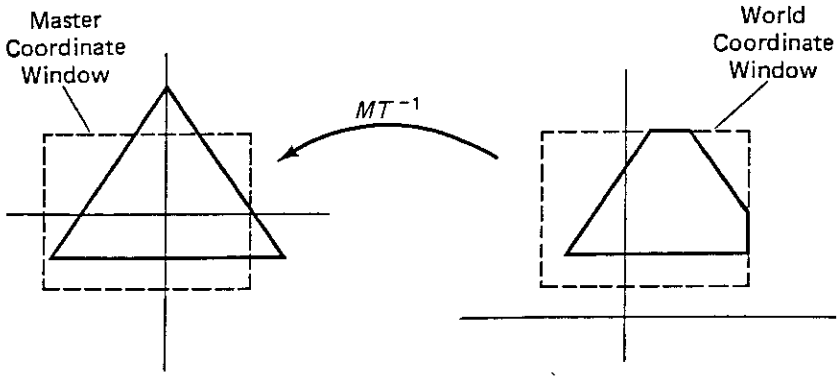
Combining Modeling and Viewing Transformations



- **master coordinate clipping**
  - **clipping can be performed before any transformations are applied**



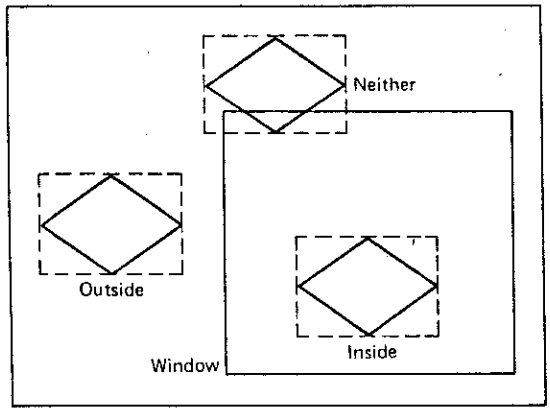
  - **when neither the symbol nor the window is rotated, the window can be transformed conveniently to the master coordinate space of each symbol**



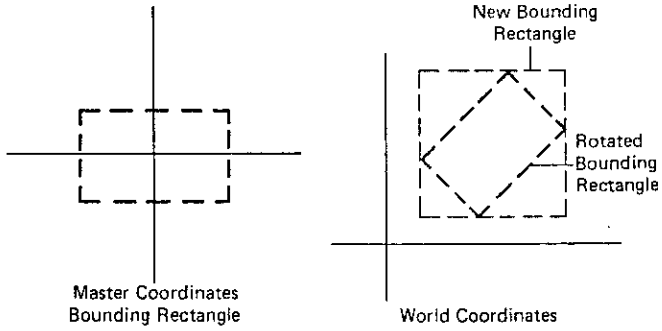
  - **master coordinate spaces don't overlap**

bounding rectangles for symbols

- bounding rectangles can be used to accept or reject symbols and modules wholesale



- bounding rectangles can be found for rotated windows and rotated symbols for wholesale acceptance or rejection



MODELING METHODS

- Basic Modeling Concepts
- Master Coordinates and Modeling Transformations
- Structured Display Files
- Symbol Operations
- Combining Modeling and Viewing Transformations