Geometric Models

What is a model?

- A representation of features of an abstract of concrete identity
- Allows people to visualize the entity and understand the behaviour of the entity
- A convenient means to experiment and predict

Computer graphics can be used to study many types of models:

- Geometric:
  - collections of components with well-defined geometry and often interconnections between components (e.g. architectural structures)
- Quantitative:
  - Equations describing some type of system (e.g. mathematical, economic, or chemical)
- Organizational:
  - Representations of hierarchies and taxonomies (e.g. org chart, library classification scheme)

- may allow many things to be tested more thoroughly and less dangerously
- models don’t need to be inherently geometric. There may be many different types of interpretations.

Geometric Models (continued)

- geometric models describe components with inherent geometrical properties and thus lend themselves naturally to graphical representation.

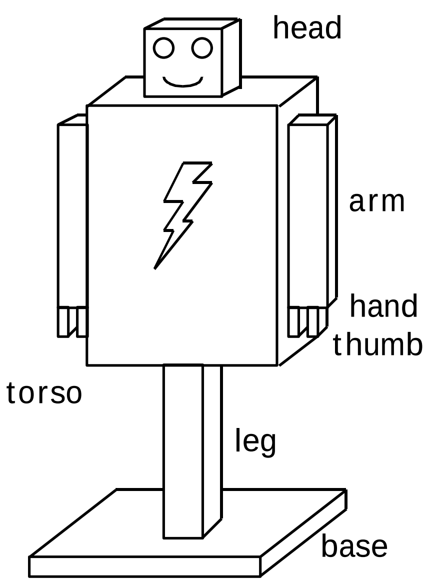
- geometric models may represent:
  - spatial layout & shape (geometry) of components and attributes such as a colour.
  - connectivity of components (topology) which may be implicit (ex: a list of points) or explicit (<a,b> connects to <c,d>).
  - application-specific data such as descriptive text

- classic space-time trade-off between what is stored explicitly and what must be computed

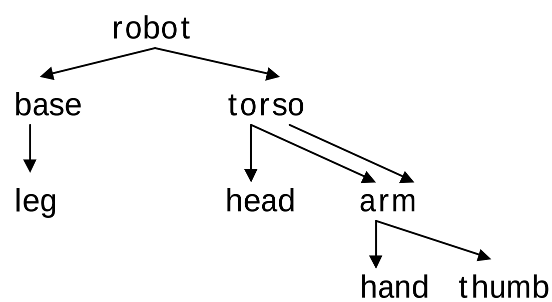
Hierarchy in Geometric Models

- geometric models often have a hierarchical structure (usually bottom-up)
- components are used as building blocks to create higher-level entities... (bottom-up)
- may also decompose components into lower-level entities... (top-down)
- the hierarchy is symbolized by a DAG (directed acyclic graph)

ex: a simple robot

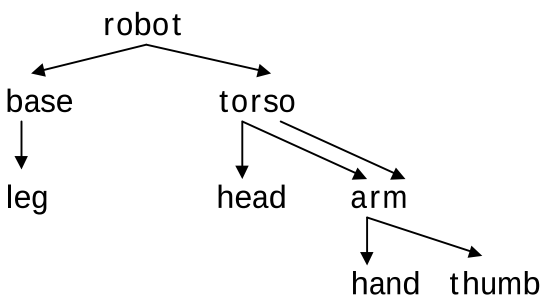


RoboDude

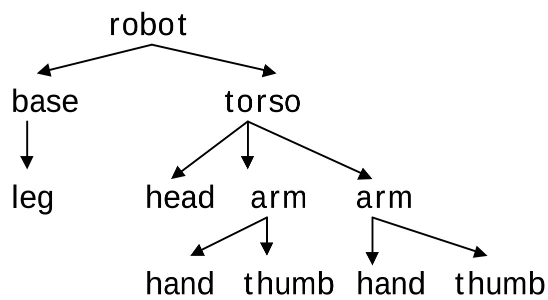


RoboDude's DAG

Comparisons to tree structures

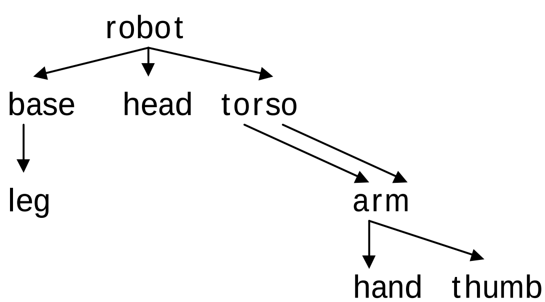


DAG



TREE

Robot may be represented by other DAGs...

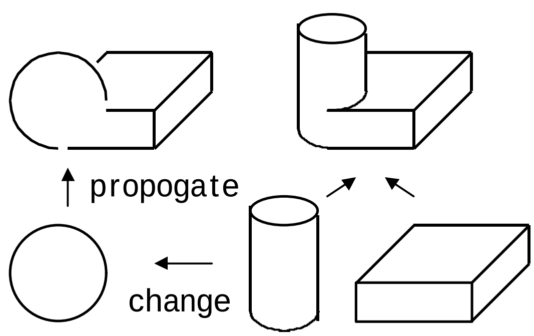


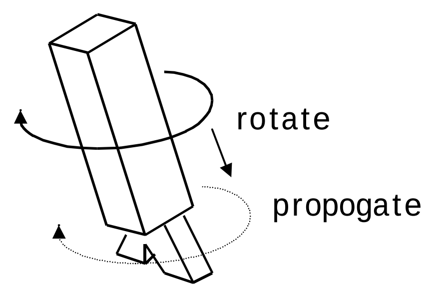
A hierarchy is created for a variety of purposes:

- to construct complex objects in a modular fashion
- to increase storage economy (store references to "base" (library) of primitive objects, such as cubes, spheres, cylinders...)
- to allow easy update propagation (a change in the definition of one building-block is automatically propagated to all of the higher-level, and, in some-cases, to levels under the object).

Examples:

1. propagate to high-level (solid modelling)
2. to lower level (transformations)





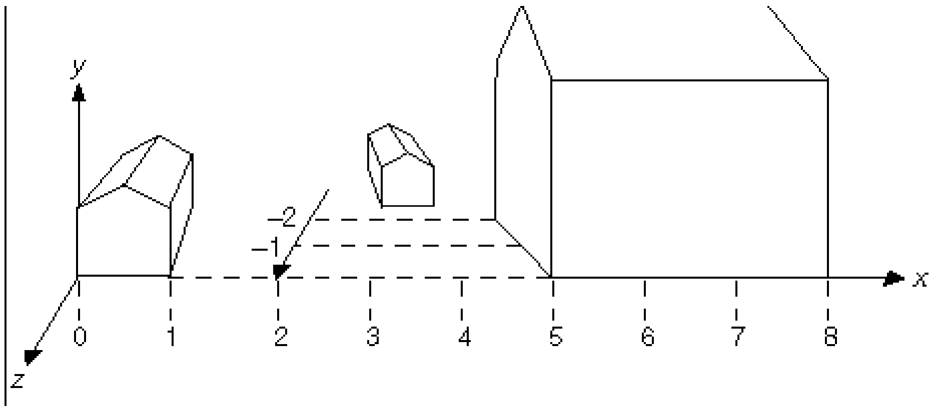
Modeling Transformations

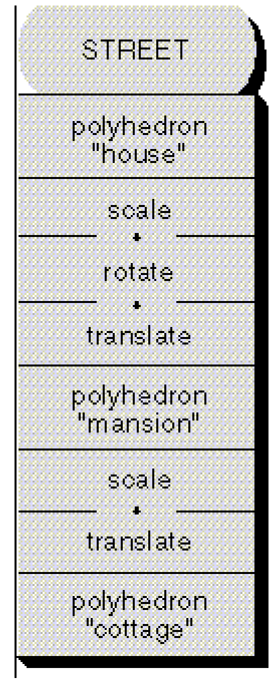
- Standardized objects:
    - those defined at the origin and largely aligned with the principal axes, are useful because they are easy to define and manipulate

- Transformations can be applied locally (to only the current object) or more globally (to all those objects which follow it).

- OpenGL allows this behaviour with its matrix stack and the glPushMatrix/glPopMatrix commands.

One-Level Hierarchy

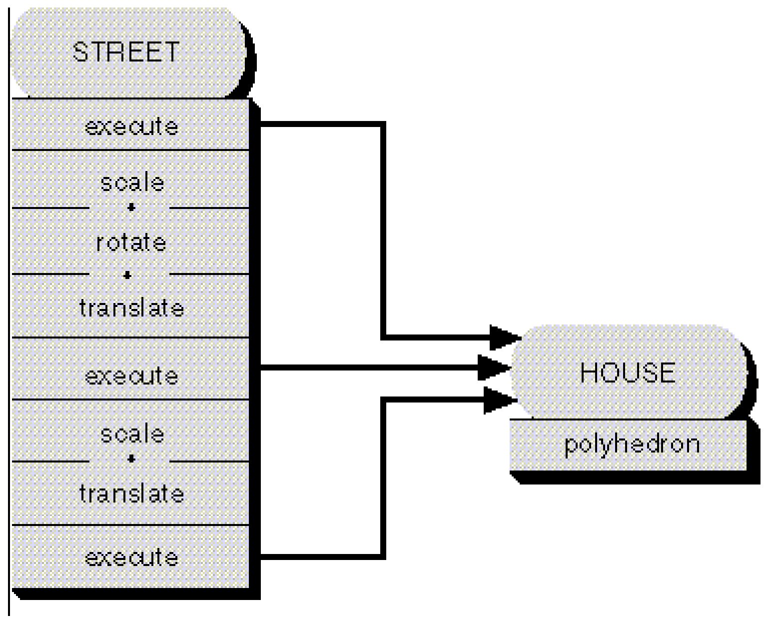





```
SPH_openStructure(STREET_STRUCT);
  /* Define first house, in standard form */
  SPH_polyhedron(…);

  /* Mansion is scaled by 2 in X, by 3 in y, and by 1 in z,
   rotated 90° about y, then translated. Note that its left side
   is subsequently front-facing and lies in the (x,y) plane. */
  SPH_setLocalTransformation(SPH_scale(2.0,3.0,2.0), REPLACE);
  SPH_setLocalTransformation(SPH_rotateY(90), PRECONCATENATE);
  SPH_setLocalTransformation(SPH_translate(8.0,0.0,0.0),
       PRECONCATENATE);
  SPH_polyhedron(…);

  /*Cottage is uniformly scaled by 0.75, unrotated, set back in
    z and over in x */
  SPH_setLocalTransformation(SPH_scale(0.75,0.75,0.75), REPLACE);
  SPH_setLocalTransformation(SPH_translate(3.5,0.0,-2.5),
           PRECONCATENATE);
  SPH_polyhedron(…);
SPH_closeStructure();
```

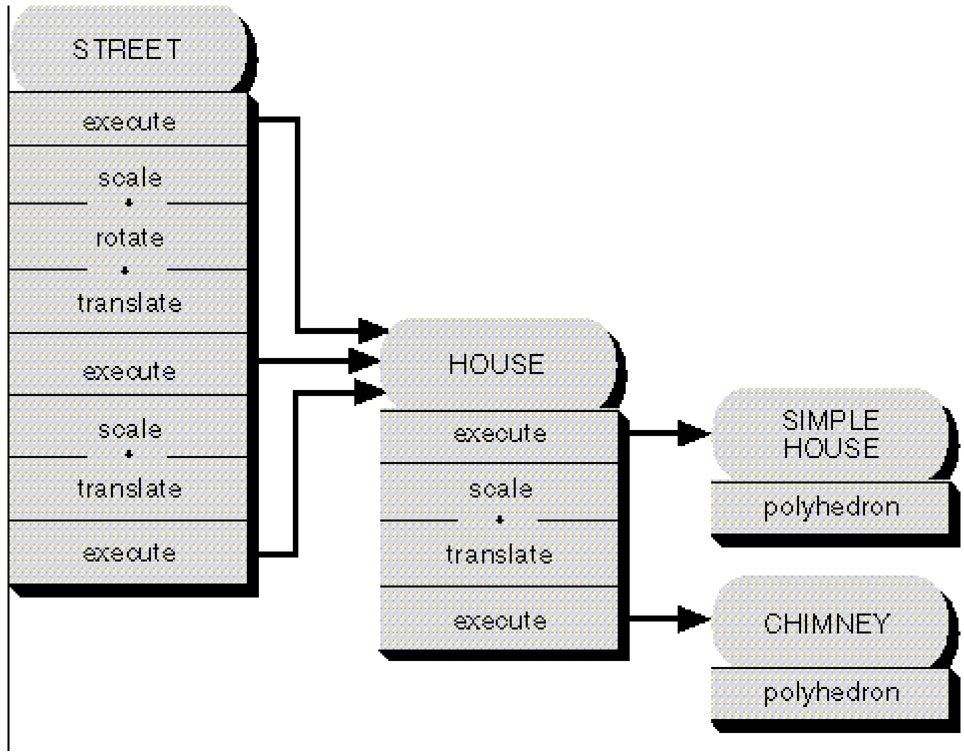
- Could also define a function to draw the house (template function)

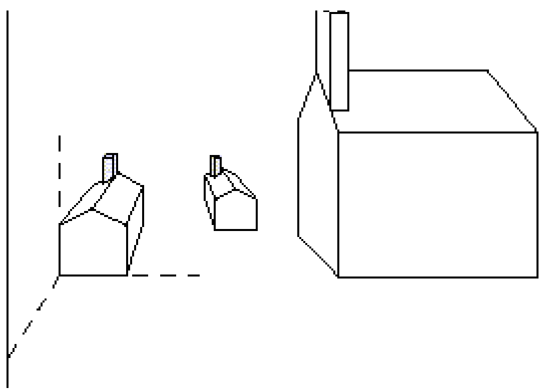
Two-level Hierarchy



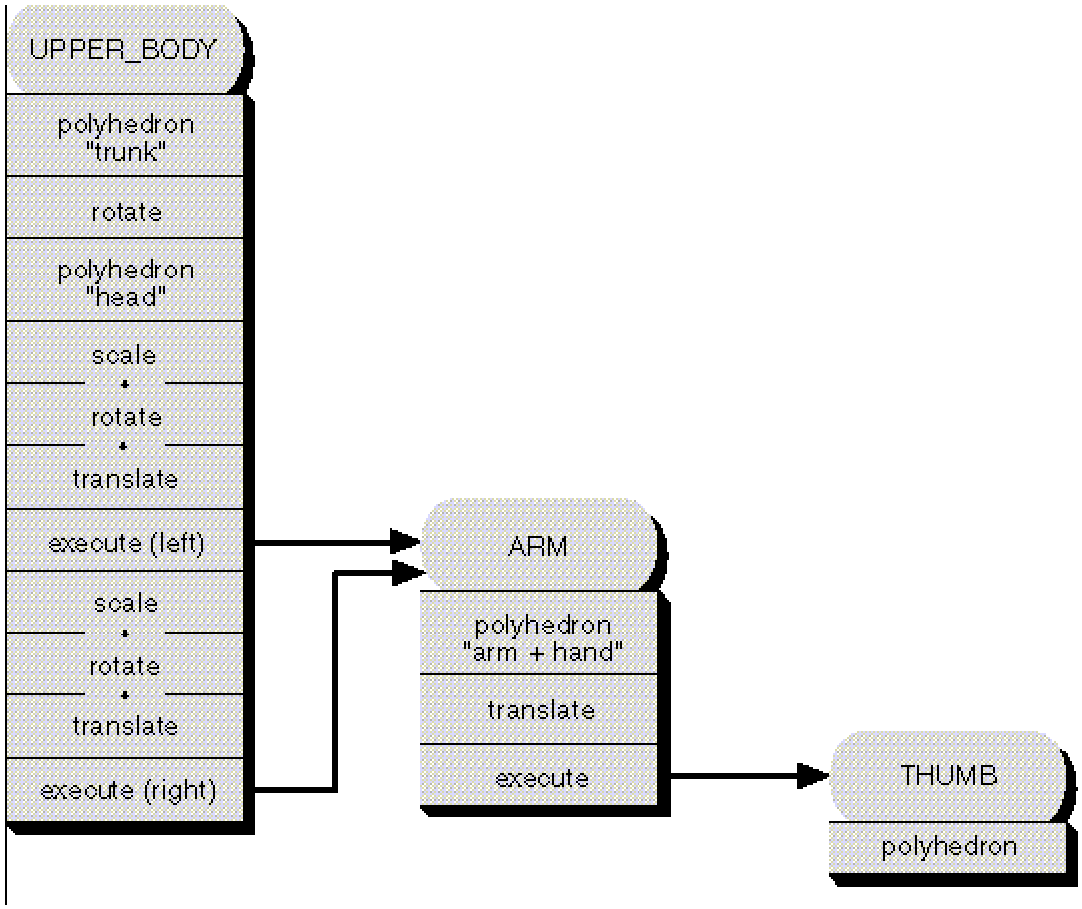

```
Void BuildStandardizedHouse
{
  SPH_openStructure(HOUSE_STRUCT);
    SPH_polyhedron(…);
  SPH_closeStructure();
}


main()
{
  BuildStandardizedHouse();

  SPH_openStructure(STREET_STRUCT);
    SPH_executeStructure(HOUSE_STRUCT);
    set local transformation matrix
    SPH_executeStructure(HOUSE_STRUCT);
    set local transformation matrix
    SPH_executeStructure(HOUSE_STRUCT);
  SPH_closeStructure();
```

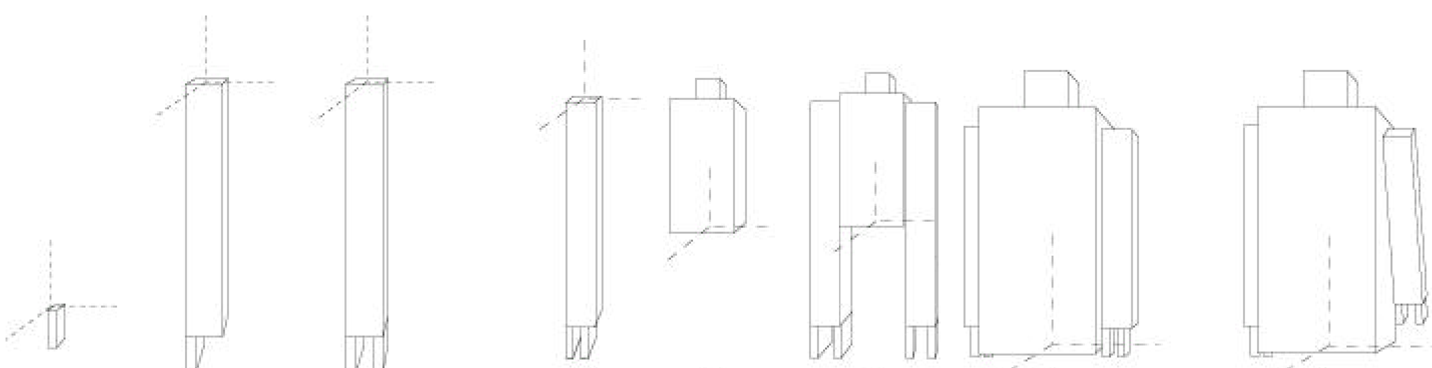
Three-level Hierarchy





Bottom-up Construction of the Robot





(a) (b) (c) (d) (e) (f) (g) (h)

Inheritance Rules