Modelling

- To specify an object to display, we need to specify the primitives making up the object.
- Typically we use model coordinate space.
- If we wish to create many different objects of a particular model, we simply create instances of the model.
- Use transformations to scale, orient, and place the instance—each instance has its own transformation matrices $M = TRS$.
- Optional: object-oriented techniques can be used to create instances.

Hierarchical Models

- Sometimes objects are placed in relation to other objects to form larger objects.
- e.g. a car has a body, wheels, windows, etc.
- e.g. a robot has a body, a head, arms, and legs.
- It would be more convenient to define objects parts relative to another. For example, if the body moves then anything attached to the body also moves by the same amount.

Hierarchical Models

- Each object has a main part (e.g. body of a robot). The object has certain coordinates in the model coordinate space.
- We can define each part in its own model space, and use a transformation matrix $M = TRS$ to attach it to the main part.
- Parts can be attached to other parts as well (e.g. upper arm to body, forearm to upper arm, hand to upper arm, etc.)
- The hierarchy can be represented as a rooted tree (root = main part).
- Each node contains the part's vertices in its own model space, together with transformation matrix to place it relative to its parent.
- The transformation matrix at the root places the entire object (all its parts) at the correct location.

Hierarchical Models

- To place a particular part in world coordinates, we must multiply the part's vertices by all transformation matrices from the root of the tree to the node.
- If $M_1, \ldots, M_p$ are the transformation matrices from root to node p, then the final coordinates of part p is multiplied by the matrix $M_1 \cdots M_p$.

Rendering Hierarchical Models

- We can use pre-order traversal of a tree.
- We maintain a current transformation matrix C that is used to multiply the vertices in the current node.
- When we enter a node in pre-order traversal, we multiply C by the node's transformation matrix (on the right).
- The matrix C is used to transform the vertices in current node.
- Before we recurse to subtrees, we must save current transformation matrix—when that subtree finishes we have to restore the current transformation matrix for other subtrees.
- We can handle this explicitly with our own stack, or we can make use of local variables in recursion.

Drawing Hierarchical Models

Pseudocode:

```
void render(Matrix C, Node *root)
{
  if (empty tree) return;
  C *= root->M;
  vertices in node root are transformed by C
  for each child of root
    render(C, child)
}
```

Robotic Arm

- Example: a 3-segment robotic arm in 2-dimension
- Each segment is a simple rectangle of unit length
- First segment is attached to origin.
- Other segments are attached to previous segment.
- Each segment is defined by length (scale), angle with previous segment (rotation) and position (translation).
- Arm configuration defined by angles at each joint: affects rotation in each node.
- To render each segment, multiply matrices from first segment to that segment.