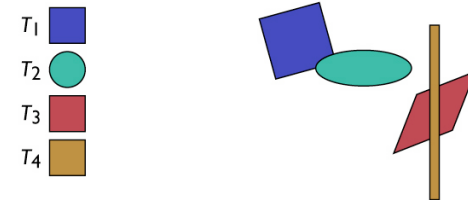


Scene Graphs

CS 4620 Lecture 6

Data structures with transforms

- Representing a drawing (“scene”)
- List of objects
- Transform for each object
 - can use minimal primitives: ellipse is transformed circle
 - transform applies to points of object



Example

- Can represent drawing with flat list
 - but editing operations require updating many transforms

T1 • □ T2 • ▲ T3 • ■ T4 • ■ T5 • ■ T6 • ■ T7 • ○ T8 • ■ T9 • ■ T10 • ■ T11 • ■ T12 • ■ T13 • ■ T14 • ■ T15 • ■ T16 • ■ T17 • ■ T18 • ■ ...



Example

- Can represent drawing with flat list
 - but editing operations require updating many transforms

T1 • □ T2 • ▲ T3 • ■ T4 • ■ T5 • ■ T6 • ■ T7 • ○ T8 • ■ T9 • ■ T10 • ■ T11 • ■ T12 • ■ T13 • ■ T14 • ■ T15 • ■ T16 • ■ T17 • ■ T18 • ■ ...

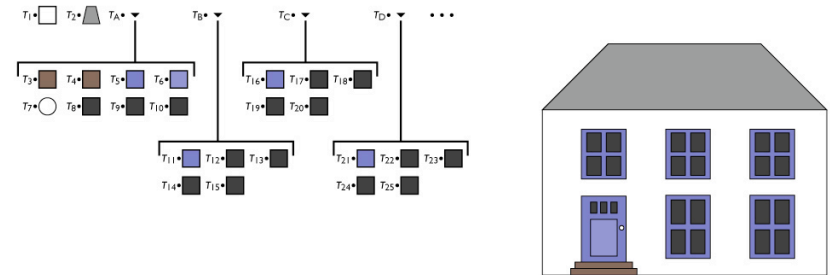


Groups of objects

- Treat a set of objects as one
- Introduce new object type: group
 - contains list of references to member objects
- This makes the model into a tree
 - interior nodes = groups
 - leaf nodes = objects
 - edges = membership of object in group

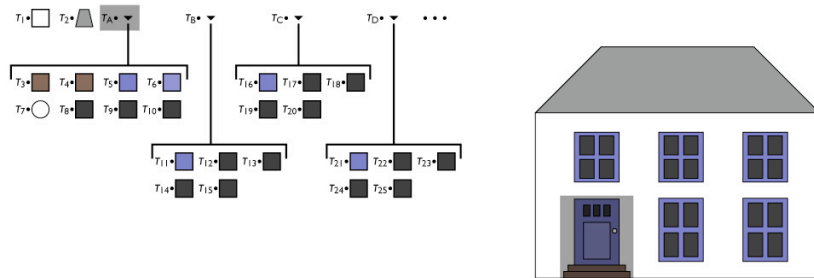
Example

- Add group as a new object type
 - lets the data structure reflect the drawing structure
 - enables high-level editing by changing just one node



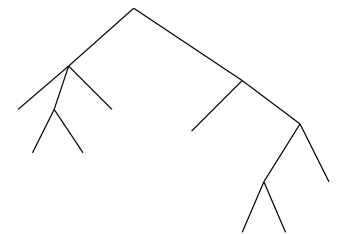
Example

- Add group as a new object type
 - lets the data structure reflect the drawing structure
 - enables high-level editing by changing just one node



The Scene Graph (tree)

- A name given to various kinds of graph structures (nodes connected together) used to represent scenes
- Simplest form: tree
 - just saw this
 - every node has one parent
 - leaf nodes are identified with objects in the scene



Concatenation and hierarchy

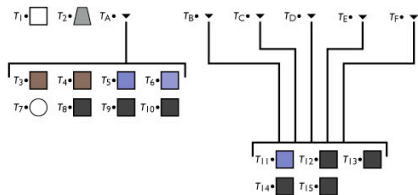
- Transforms associated with nodes or edges
- Each transform applies to all geometry below it
 - want group transform to transform each member
 - members already transformed—concatenate
- Frame transform for object is product of all matrices along path from root
 - each object's transform describes relationship between its local coordinates and its group's coordinates
 - frame-to-canonical transform is the result of repeatedly changing coordinates from group to containing group

Instances

- Simple idea: allow an object to be a member of more than one group at once
 - transform different in each case
 - leads to linked copies
 - single editing operation changes all instances

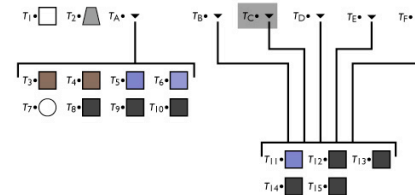
Example

- Allow multiple references to nodes
 - reflects more of drawing structure
 - allows editing of repeated parts in one operation



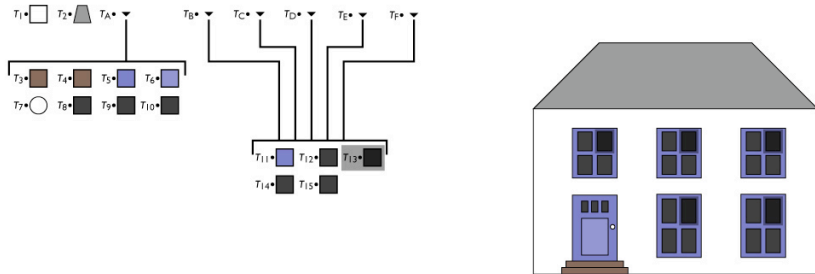
Example

- Allow multiple references to nodes
 - reflects more of drawing structure
 - allows editing of repeated parts in one operation



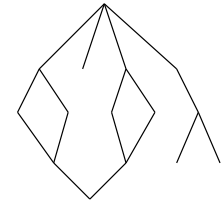
Example

- Allow multiple references to nodes
 - reflects more of drawing structure
 - allows editing of repeated parts in one operation



The Scene Graph (with instances)

- With instances, there is no more tree
 - an object that is instanced multiple times has more than one parent
- Transform tree becomes DAG
 - directed **acyclic** graph
 - group is not allowed to contain itself, even indirectly
- Transforms still accumulate along path from root
 - now *paths* from root to leaves are identified with scene objects



Implementing a hierarchy

- Object-oriented language is convenient
 - define shapes and groups as derived from single class

```
abstract class Shape {
    void draw();
}

class Square extends Shape {
    void draw() {
        // draw unit square
    }
}

class Circle extends Shape {
    void draw() {
        // draw unit circle
    }
}
```

Implementing traversal

- Pass a transform down the hierarchy
 - before drawing, concatenate

```
abstract class Shape {
    void draw(Transform t_c);
}

class Square extends Shape {
    void draw(Transform t_c) {
        // draw t_c * unit square
    }
}

class Circle extends Shape {
    void draw(Transform t_c) {
        // draw t_c * unit circle
    }
}
```

Implementing traversal

- Pass a transform down the hierarchy
 - before drawing, concatenate

```
abstract class Shape {
    void draw(Transform t_c);
}

class Square extends Shape {
    void draw(Transform t_c) {
        // draw t_c * unit square
    }
}

class Circle extends Shape {
    void draw(Transform t_c) {
        // draw t_c * unit circle
    }
}

class Group extends Shape {
    Transform t;
    ShapeList members;
    void draw(Transform t_c) {
        for (m in members) {
            m.draw(t_c * t);
        }
    }
}
```

Basic Scene Graph operations

- Editing a transformation
 - good to present usable UI
- Getting transform of object in canonical (world) frame
 - traverse path from root to leaf
- Grouping and ungrouping
 - can do these operations without moving anything
 - group: insert identity node
 - ungroup: remove node, push transform to children
- Reparenting
 - move node from one parent to another
 - can do without altering position

Adding more than geometry

- Objects have properties besides shape
 - color, shading parameters
 - approximation parameters (e.g. precision of subdividing curved surfaces into triangles)
 - behavior in response to user input
 - ...
- Setting properties for entire groups is useful
 - paint entire window green
- Many systems include some kind of property nodes
 - in traversal they are read as, e.g., “set current color”

Scene Graph variations

- Where transforms go
 - in every node
 - on edges
 - in group nodes only
 - in special Transform nodes
- Tree vs. DAG
- Nodes for cameras and lights?