OpenGL: hierarchical objects

Queen Mary University of London

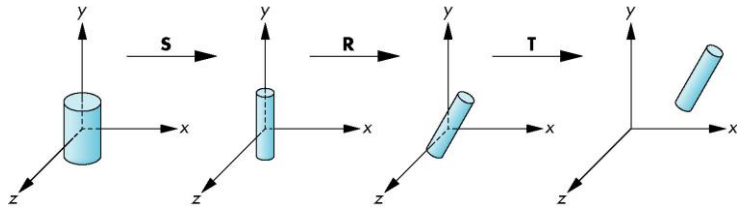
Objectives

- Examine the limitations of linear modelling
  - Symbols and instances
- Introduce hierarchical models
  - Articulated models
  - Robots

Queen Mary University of London

Instance Transformation

- Start with a prototype object (a *symbol*)
- Each appearance of the object in the model is an *instance*
  - Must scale, orient, position
  - Defines instance transformation

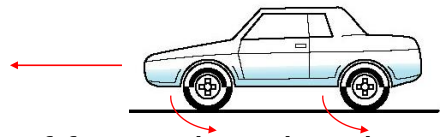


Queen Mary University of London

3

Relationships in Car Model

- Consider model of car
  - Chassis + 4 identical wheels
  - Two symbols



- Rate of forward motion determined by rotational speed of wheels

Queen Mary University of London

4

Structure Through Function Calls

```
car(speed)
{
     chassis()
     wheel(right_front);
     wheel(left_front);
     wheel(right_rear);
     wheel(left_rear);
}
```

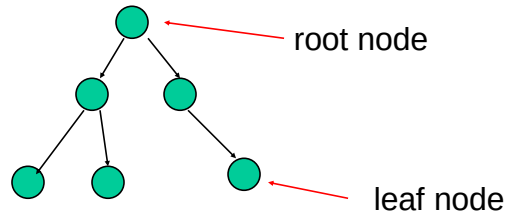
- Fails to show relationships well
- Need to specify the position of each wheel independently

Queen Mary University of London

5

Tree

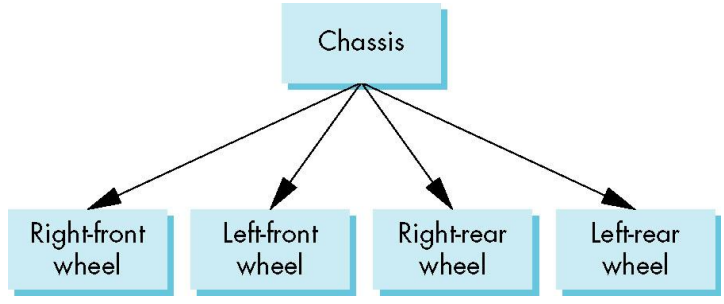
- Graph in which each node (except the root) has exactly one parent node
  - May have multiple children
  - Leaf or terminal node: no children



Queen Mary University of London

6

Tree Model of Car

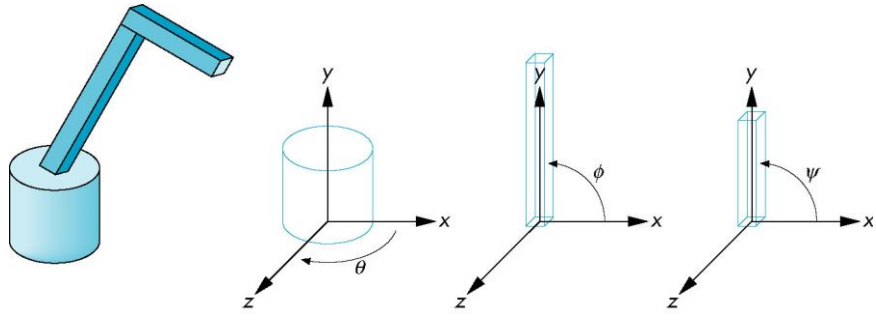


Allows to define the transform of the child with respect to the transform of the parent

Queen Mary University of London

7

Robot Arm



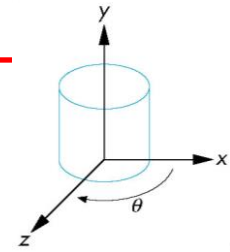
robot arm

parts in their own coordinate systems

Queen Mary University of London

8

base()

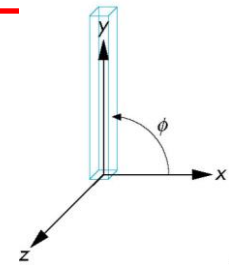



```
#define BASE_RADIUS 1.0
#define BASE_HEIGHT 1.5

GLUquadricObj *p;   // pointer to quadric object

void base()
{
 glPushMatrix();
 glRotatef(-90.0, 1.0, 0.0, 0.0);
 gluCylinder(p,BASE_RADIUS,BASE_RADIUS,BASE_HEIGHT,15,15);
 glPopMatrix();
}
```

Queen Mary University of London

9

lower_arm()

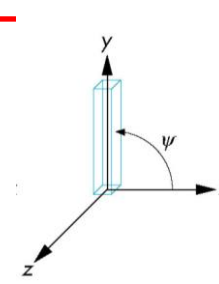



```
#define LOWER_ARM_WIDTH  0.5
#define LOWER_ARM_HEIGHT 3.5

void lower_arm()
{
 glPushMatrix();
 glTranslatef(0.0, 0.5*LOWER_ARM_HEIGHT, 0.0);
 glScalef(LOWER_ARM_WIDTH,LOWER_ARM_HEIGHT,LOWER_ARM_WIDTH);
 glutWireCube(1.0);
 glPopMatrix();
}
```

Queen Mary University of London

10

upper_arm()



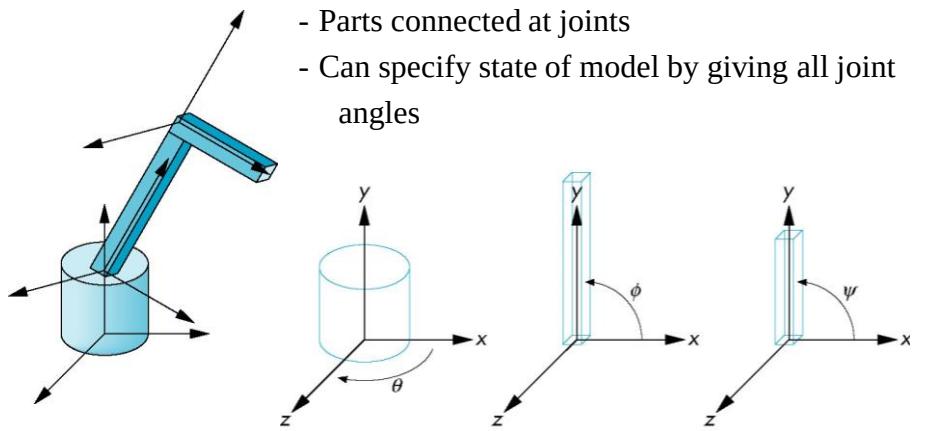

```
#define UPPER_ARM_WIDTH  0.3
#define UPPER_ARM_HEIGHT 2.0

void upper_arm()
{
 glPushMatrix();
 glTranslatef(0.0, 0.5*UPPER_ARM_HEIGHT, 0.0);
 glScalef(UPPER_ARM_WIDTH,UPPER_ARM_HEIGHT,UPPER_ARM_WIDTH);
 glutWireCube(1.0);
 glPopMatrix();
}
```

Queen Mary University of London

11

Articulated Models

- The robot arm is an example of an *articulated model*
  - Parts connected at joints
  - Can specify state of model by giving all joint angles



Queen Mary University of London

12

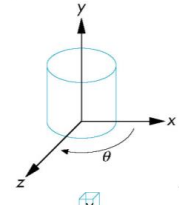
Relationships in Robot Arm

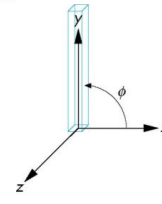
- Base rotates independently
  - Single angle determines position
- Lower arm attached to base
  - Its position depends on rotation of base
  - Must also translate relative to base and rotate about connecting joint
- Upper arm attached to lower arm
  - Its position depends on both base and lower arm
  - Must translate relative to lower arm and rotate about joint connecting to lower arm

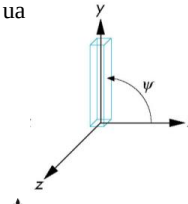
Queen Mary University of London

13

Required Matrices





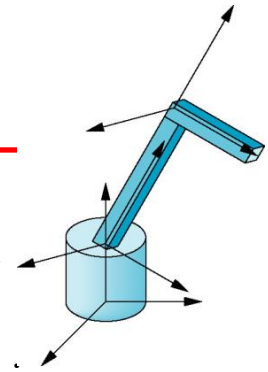


- Rotation of base (y axis): $\mathbf{R}_b$
  - Apply $\mathbf{M} = \mathbf{R}_b$ to base
- Translate lower arm relative to base: $\mathbf{T}_{la}$
- Rotate lower arm around joint (z axis): $\mathbf{R}_{la}$
  - Apply $\mathbf{M} = \mathbf{R}_b \mathbf{T}_{la} \mathbf{R}_{la}$ to lower arm
- Translate upper arm relative to lower arm: $\mathbf{T}_{ua}$
- Rotate upper arm around joint (z axis): $\mathbf{R}_{ua}$
  - Apply $\mathbf{M} = \mathbf{R}_b \mathbf{T}_{la} \mathbf{R}_{la} \mathbf{T}_{ua} \mathbf{R}_{ua}$ to upper arm

Queen Mary University of London

14

robot_arm()



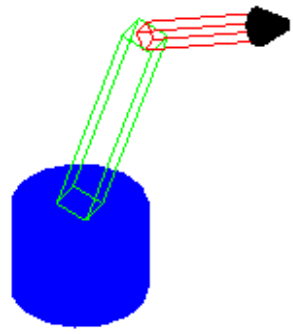
```
robot_arm()
{
 glRotatef(theta[0], 0.0, 1.0, 0.0);
 base();

 glTranslatef(0.0, BASE_HEIGHT, 0.0);
 glRotatef(theta[1], 0.0, 0.0, 1.0);
 lower_arm();

 glTranslatef(0.0, LOWER_ARM_HEIGHT, 0.0);
 glRotatef(theta[2], 0.0, 0.0, 1.0);
 upper_arm();
}
```

Queen Mary University of London

15

Exercise ..

Attach a "hand" at the end of the upper arm, using the gluCylinder function.

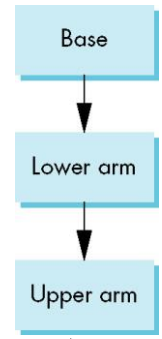


Queen Mary University of London

16

Tree Model of Robot

- Note code shows relationships between parts of model
  - Can change “look” of parts easily without altering relationships
- Simple example of tree model
- Want a general node structure for nodes



Queen Mary University of London

17

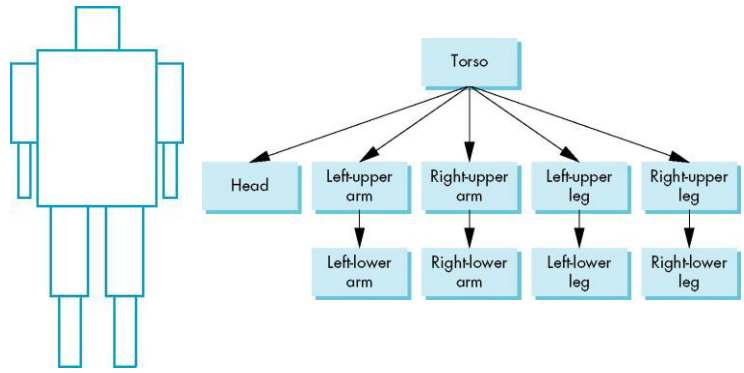
Generalizations

- Need to deal with multiple children
  - How do we represent a more general tree?
  - How do we traverse such a data structure?
- Animation
  - How to use dynamically?
  - Can we create and delete nodes during execution?

Queen Mary University of London

18

Humanoid Figure



Queen Mary University of London

19

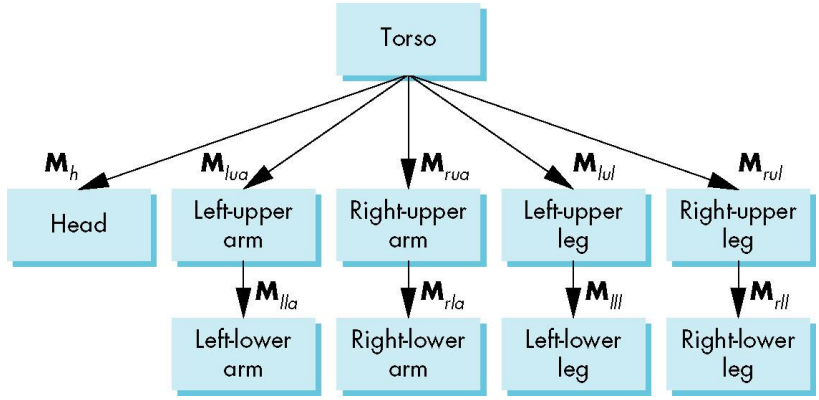
Building the Model

- Can build a simple implementation using quadrics: ellipsoids and cylinders
- Access parts through functions, e.g.:
  - `torso()`
  - `upper_arm()`
- Matrices describe position of node with respect to its parent, e.g.:
  - $\mathbf{M}_{lla}$ positions left lower arm with respect to left upper arm

Queen Mary University of London

20

Tree with Matrices



Queen Mary University of London

21

Display and Traversal

- The position of the figure is determined by 11 joint angles (two for the head and one for each other part)
- Display of the tree requires a *graph traversal*
  - Visit each node once
  - Display function at each node that describes the part associated with the node, applying the correct transformation matrix for position and orientation

Queen Mary University of London

22

Transformation Matrices

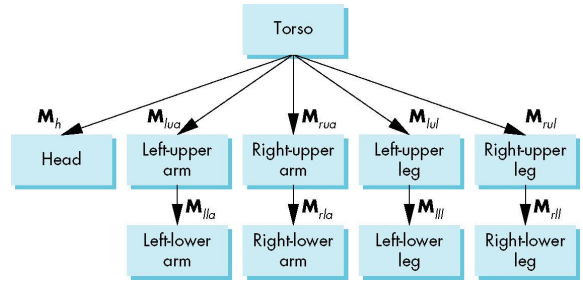
- There are 10 relevant matrices
  - $\mathbf{M}$ positions and orients entire figure through the torso which is the root node
  - $\mathbf{M}_h$ positions head with respect to torso
  - $\mathbf{M}_{lua}$, $\mathbf{M}_{rua}$, $\mathbf{M}_{lul}$, $\mathbf{M}_{rul}$ position arms and legs with respect to torso
  - $\mathbf{M}_{lla}$, $\mathbf{M}_{rla}$, $\mathbf{M}_{lll}$, $\mathbf{M}_{rll}$ position lower parts of limbs with respect to corresponding upper limbs

Queen Mary University of London

23

Stack-based Traversal

- Set model-view matrix to $\mathbf{M}$ and draw torso
- Set model-view matrix to $\mathbf{M}\mathbf{M}_h$ and draw head
- For left-upper arm need $\mathbf{M}\mathbf{M}_{lua}$ and so on
- Rather than recomputing $\mathbf{M}\mathbf{M}_{lua}$ from scratch or using an inverse matrix, we can use the matrix stack to store $\mathbf{M}$ and other matrices as we traverse the tree



Queen Mary University of London

24

Traversal Code

```
figure() {
    glPushMatrix()
    torso();
    glTranslate3f(…);
    glRotate3f(…);
    head();
    glPopMatrix();
    glPushMatrix();
    glTranslate3f(…);
    glRotate3f(…);
    upper_arm();
    glTranslate3f(…);
    glRotate3f(…);
    lower_arm();
    glPopMatrix();
    glPushMatrix();
```

save present model-view matrix

update model-view matrix for head

recover original model-view matrix

save it again

update model-view matrix for left upper arm

update model-view matrix for left lower arm

recover and save original model-view matrix again

rest of code …

Queen Mary University of London

Moving the whole body

```
figure() {
    glTranslatef();
    glRotate3f();
      …
    glPushMatrix()
    torso();
    glTranslate3f(…);
    glRotate3f(…);
    head();
```

Move the whole body

save present model-view matrix

rest of code

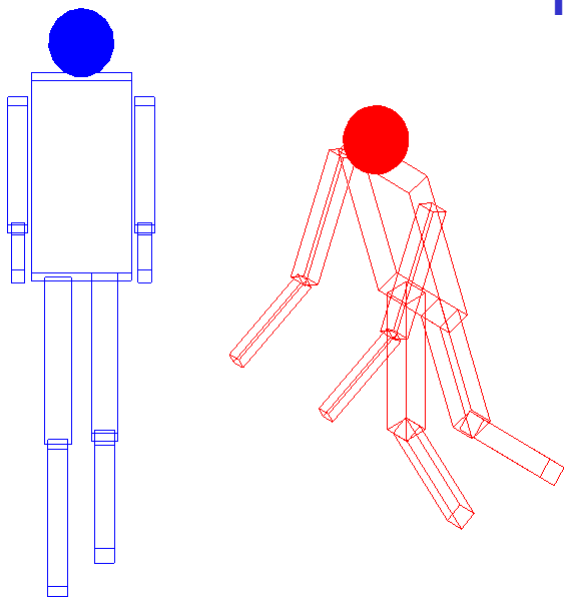
Queen Mary University of London

State changes

- Note that the sample code does not include state changes, such as changes to colors
  - May also want to use **`glPushAttrib`** and **`glPopAttrib`** to protect against unexpected state changes affecting later parts of the code
  - e.g. glPushAttrib (GL_COLOR_BUFFER_BIT);
    glPopAttrib();

Queen Mary University of London

Humanoid …



Queen Mary University of London