

Hierarchical Modeling

Tuesday, October 29, 2013

Approaches & Transformation Stacks

- Polygonal Meshes (review)
- Constructive Solid Geometry
- Splines
- A relative to B relative to C ...
 - Lecture notes draw upon OpenGL – legacy
 - Online example uses JavaScript Canvas

Transformation Stack

OpenGL for example

- The OpenGL state machine maintains two matrices
 - the MODELVIEW matrix
 - the PROJECTION matrix
- The transformation applied to data is the product of these two matrices.

$$\begin{bmatrix} p_1 & p_2 & p_3 & p_4 \\ p_5 & p_6 & p_7 & p_8 \\ p_9 & p_{10} & p_{11} & p_{12} \\ p_{13} & p_{14} & p_{15} & 1 \end{bmatrix} \begin{bmatrix} m_1 & m_2 & m_3 & m_4 \\ m_5 & m_6 & m_7 & m_8 \\ m_9 & m_{10} & m_{11} & m_{12} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

OpenGL Transformations (II)

- You can manipulate these matrices directly
- But I really don't recommend this approach

```
GLfloat my_matrix1[4][4] = {...}, {...}, ...};
GLfloat my_matrix2[4][4] = {...}, {...}, ...};
glMatrixMode(GL_MODELVIEW);
glLoadMatrix(&my_matrix1);
glMultMatrix(&my_matrix2);
```

Transformations (III)

- A much better approach.
- Start with a known matrix
 - Identity is nice!
- Then apply affine transformations to it

```
glMatrixMode(GL_MODELVIEW);  
glLoadIdentity();  
glTranslatef(dx, dy, dz);  
glScalef(sx, sy, sz);
```

Under the Hood

- What do these routines do?

```
// glTranslatef(dx, dy, dz)  
GLfloat trans = {{1.0, 0.0, 0.0, dx},  
                {0.0, 1.0, 0.0, dy},  
                {0.0, 0.0, 1.0, dz},  
                {0.0, 0.0, 0.0, 1.0}};  
glMultMatrix(trans);
```

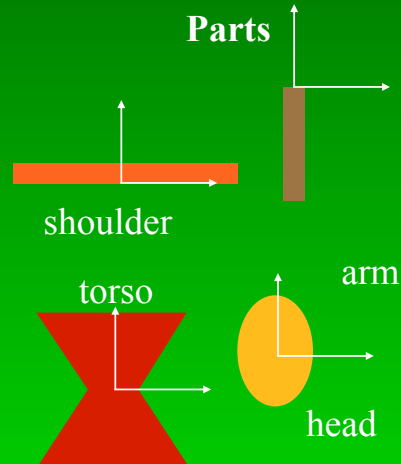
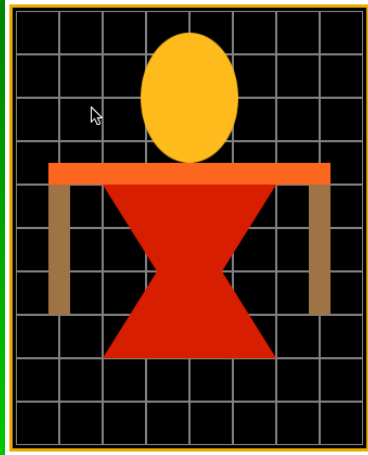
Matrix Stacks

- General concept – realized in many APIs
- OpenGL MODELVIEW is a matrix stack
 - When you push a new matrix,
 - the top of the stack is replicated.
 - Now modify the current coordinate frame
 - Pop the stack
 - Return to the original coordinate system
- The PROJECTION matrix is also a stack

Manipulating Stacks

```
/* to select which matrix stack... */  
void glMatrixMode(GL_MODELVIEW);  
  
/* to push a new matrix onto the stack */  
/* ... a copy of previous top of stack */  
void glPushMatrix();  
  
/* Manipulate new top-level matrix */  
  
/* pop new matrix off the stack */  
void glPopMatrix();
```

JavaScript Example

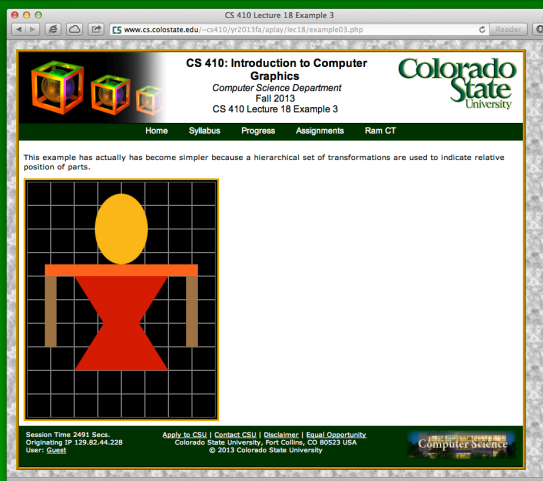


10/24/13

©Bruce A. Draper & J. Ross Beveridge 2013

9

JavaScript Example



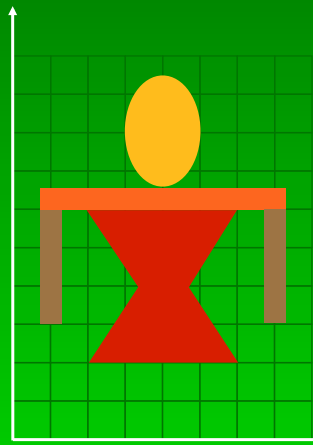
10/24/13

©Bruce A. Draper & J. Ross Beveridge 2013

10

OpenGL Example

Dependent sequence



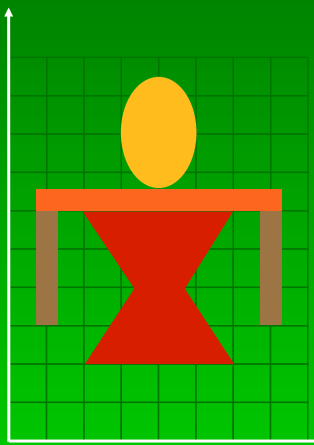
```
glMatrixMode(GL_MODELVIEW);  
glLoadIdentity();  
glTranslate(4,4,0);  
drawTorso();  
glTranslate(0,2,0);  
drawShoulder();  
glTranslate(3,0,0);  
drawArm();  
glTranslate(-6,0,0);  
drawArm();  
glTranslate(3,2,0);  
drawHead();
```

10/24/13

©Bruce A. Draper & J. Ross Beveridge 2013

11

A Bit Better - Parts Independent



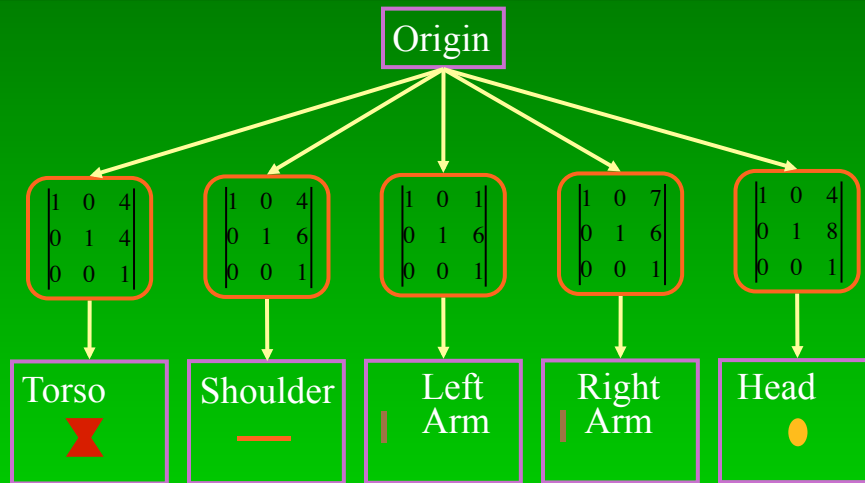
```
glMatrixMode(GL_MODELVIEW);  
glLoadIdentity();  
  
glPushMatrix();  
glTranslate(4,4);  
drawTorso();  
glPopMatrix();  
  
glPushMatrix();  
glTranslate(4,6);  
drawShoulder();  
glPopMatrix();  
  
glPushMatrix();  
glTranslate(1,6);  
drawArm();  
glPopMatrix();  
  
glPushMatrix();  
glTranslate(7,6);  
drawArm();  
glPopMatrix();  
  
glPushMatrix();  
glTranslate(4,8);  
drawHead();  
glPopMatrix();
```

10/24/13

©Bruce A. Draper & J. Ross Beveridge 2013

12

Previous example as a Graph

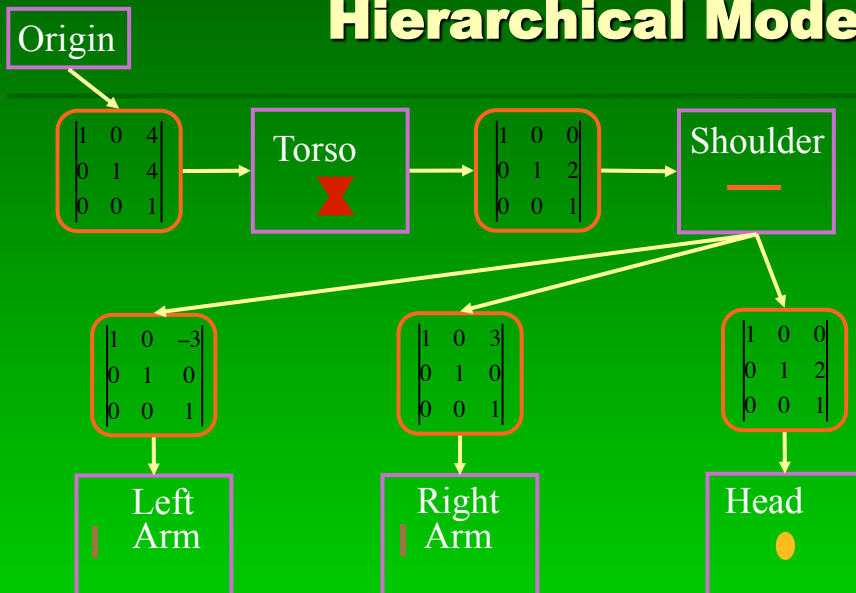


10/24/13

©Bruce A. Draper & J. Ross Beveridge 2013

13

Hierarchical Model



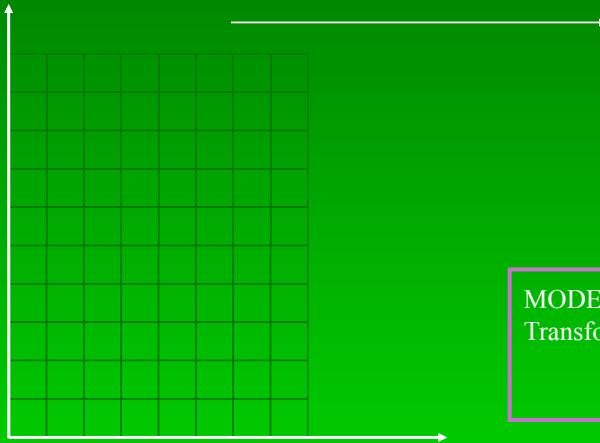
10/24/13

©Bruce A. Draper & J. Ross Beveridge 2013

14

Initialization

```
glMatrixMode(GL_MODELVIEW);  
glLoadIdentity();  
glPushMatrix();
```



Stack

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

MODELVIEW
Transformation

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

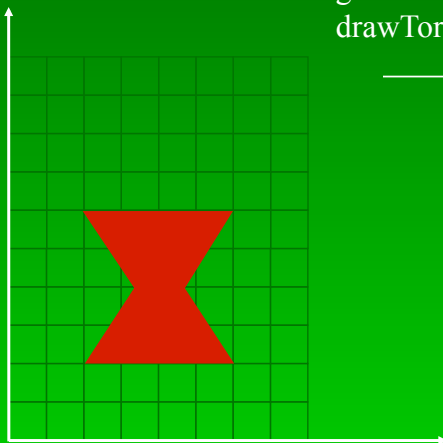
10/24/13

©Bruce A. Draper & J. Ross Beveridge 2013

15

Torso

```
glPushMatrix();  
glTranslate(4,4);  
glPushMatrix();  
drawTorso();
```



Stack

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$
$$\begin{pmatrix} 1 & 0 & 4 \\ 0 & 1 & 4 \\ 0 & 0 & 1 \end{pmatrix}$$

MODELVIEW
Transformation

$$\begin{pmatrix} 1 & 0 & 4 \\ 0 & 1 & 4 \\ 0 & 0 & 1 \end{pmatrix}$$

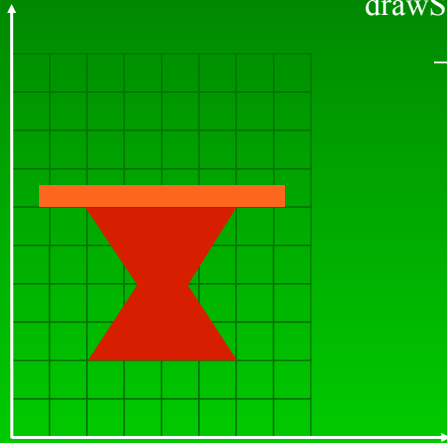
10/24/13

©Bruce A. Draper & J. Ross Beveridge 2013

16

Shoulders

```
glTranslate(0,2);  
glPushMatrix();  
drawShoulders();
```



Stack

```
| 1 0 0 |  
| 0 1 0 |  
| 0 0 1 |  
| 1 0 4 |  
| 0 1 4 |  
| 0 0 1 |
```

```
| 1 0 4 |  
| 0 1 6 |  
| 0 0 1 |
```

MODELVIEW
Transformation

```
| 1 0 4 |  
| 0 1 6 |  
| 0 0 1 |
```

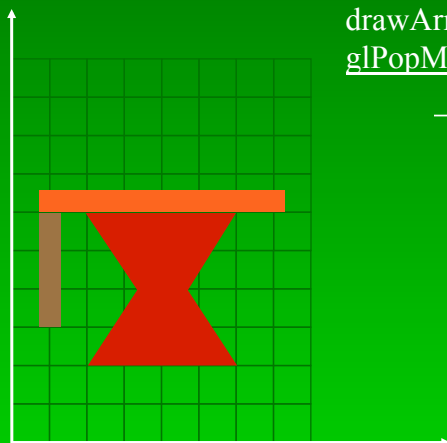
10/24/13

©Bruce A. Draper & J. Ross Beveridge 2013

17

Left Arm

```
glPushMatrix();  
gltranslate(-3,0);  
drawArm();  
glPopMatrix();
```



Stack

```
| 1 0 0 |  
| 0 1 0 |  
| 0 0 1 |  
| 1 0 4 |  
| 0 1 4 |  
| 0 0 1 |
```

```
| 1 0 4 |  
| 0 1 6 |  
| 0 0 1 |
```

MODELVIEW
Transformation

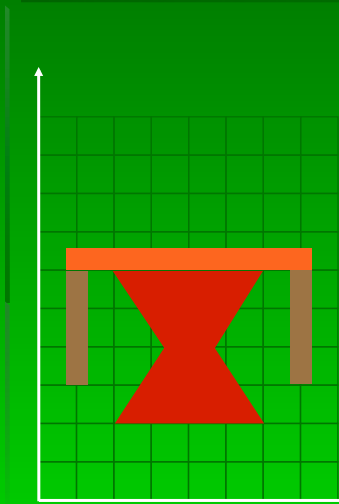
```
| 1 0 1 |  
| 0 1 6 |  
| 0 0 1 |
```

10/24/13

©Bruce A. Draper & J. Ross Beveridge 2013

18

Right Arm



```
glPushMatrix();  
gltranslate(3,0);  
drawArm();  
glPopMatrix();
```

Stack

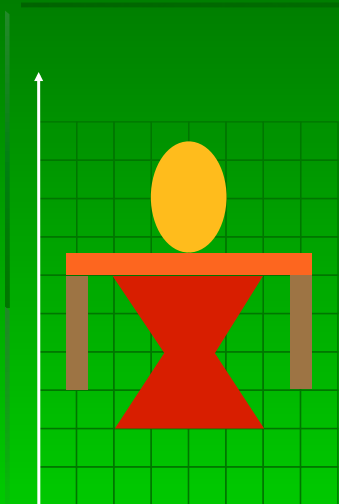
1	0	0
0	1	0
0	0	1
1	0	4
0	1	4
0	0	1

1	0	4
0	1	6
0	0	1

MODELVIEW Transformation

1	0	6
0	1	6
0	0	1

Head



```
glPushMatrix();  
gltranslate(0,2);  
drawHead();  
glPopMatrix();
```

Stack

1	0	0
0	1	0
0	0	1
1	0	4
0	1	4
0	0	1

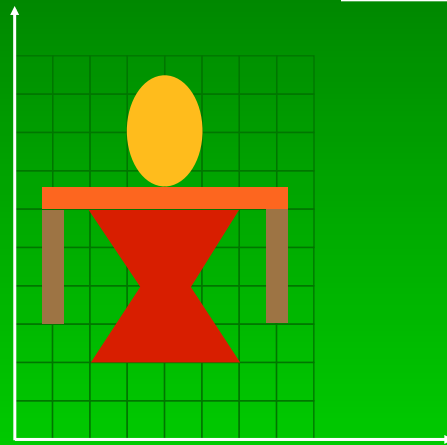
1	0	4
0	1	6
0	0	1

MODELVIEW Transformation

1	0	4
0	1	8
0	0	1

Cleanup

```
glPopMatrix();  
glPopMatrix();
```



Stack

$$\begin{vmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{vmatrix}$$

MODELVIEW
Transformation

$$\begin{vmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{vmatrix}$$

10/24/13

©Bruce A. Draper & J. Ross Beveridge 2013

21

Summary of OpenGL Code

```
glLoadIdentity();  
glPushMatrix();  
|   gltranslate(4,4);  
|   drawTorso();  
|   glPushMatrix();  
|   |   gltranslate(0,2);  
|   |   drawShoulders();  
|   |   glPushMatrix();  
|   |   |   gltranslate(-3,0);  
|   |   |   drawArm();  
|   |   glPopMatrix();  
|   |   glPushMatrix();  
|   |   |   gltranslate(3,0);  
|   |   |   drawArm();  
|   |   glPopMatrix();  
|   |   glPushMatrix();  
|   |   |   gltranslate(0,2);  
|   |   |   drawHead();  
|   |   glPopMatrix();  
|   glPopMatrix();  
glPopMatrix();
```

Indenting of code is a convenient way to indicate state of the transformation stack.

The four levels correspond with the four levels in the hierarchical model.

10/24/13

©Bruce A. Draper & J. Ross Beveridge 2013


Slide 22

CS 410 Lecture 18 Example 3

www.cs.colostate.edu/~cs410/yr2013fa/isplay/lec18/example03.php

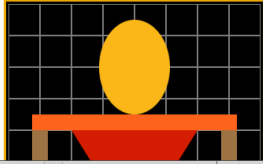
CS 410: Introduction to Computer Graphics

Computer Science Department
Fall 2013
CS 410 Lecture 18 Example 3



Home Syllabus Progress Assignments Ram CT

This example has actually become simpler because a hierarchical set of transformations are used to indicate relative position of parts.



```
7 function drawFigure(vn) {
8   ctx.save();
9   ctx.translate(4, 4);
10  drawTorso();
11
12  ctx.translate(0.0, 2.25);
13  drawShoulder();
14
15  ctx.save();
16  ctx.translate(-3.0, -1.75);
17  drawArm();
18  ctx.restore();
19
20  ctx.save();
21  ctx.translate(3.0, -1.75);
22  drawArm();
23  ctx.restore();
24
```

10/24/13 ©Bruce A. Draper & J. Ross Beveridge 2013 Slide 23