

*CS418*  
*Discussion Section*  
*(III)*  
*Arcball & VBO*

Presented by : Wei-Wen Feng  
2/11/2009

# MP1 : Mesh Rendering

- Due on February 16 (Mon)
  - Compass is sometimes not very stable. Try to submit earlier.
  - Email me if you encounter last minute failure on Compass.

# MP1

- Depth Test :  
“glEnable(GL\_DEPTH\_TEST);”
- glRotate3f :
  - OpenGL will normalize the axis.

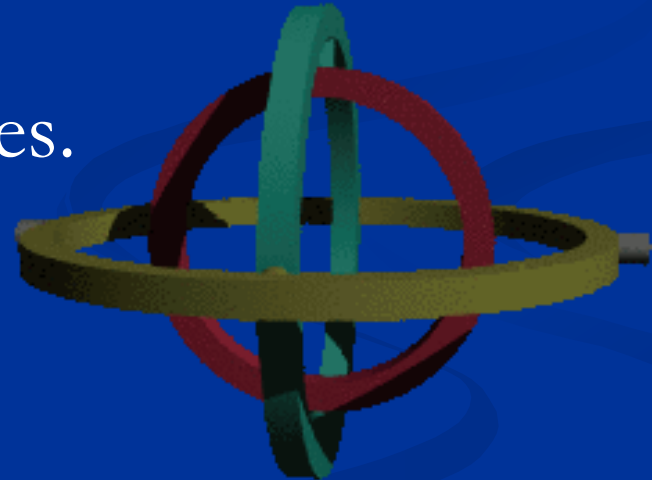
# Interactive Viewing

- Interactive viewing is desired for 3D model display.
- Adjust the orientation of shape with UI
  - FPS style : Changing the first person view → Exploring the environment
  - ArcBall ( TrackBall ) : Rotate the object at view center. → Easier to view a single object in all direction

# Euler Angles

- At most 75% of credit if you only implement Euler Angles.
- Rotate about X,Y,Z axis respectively.
- Very easy to implement.
- Keep track of X,Y,Z angles.

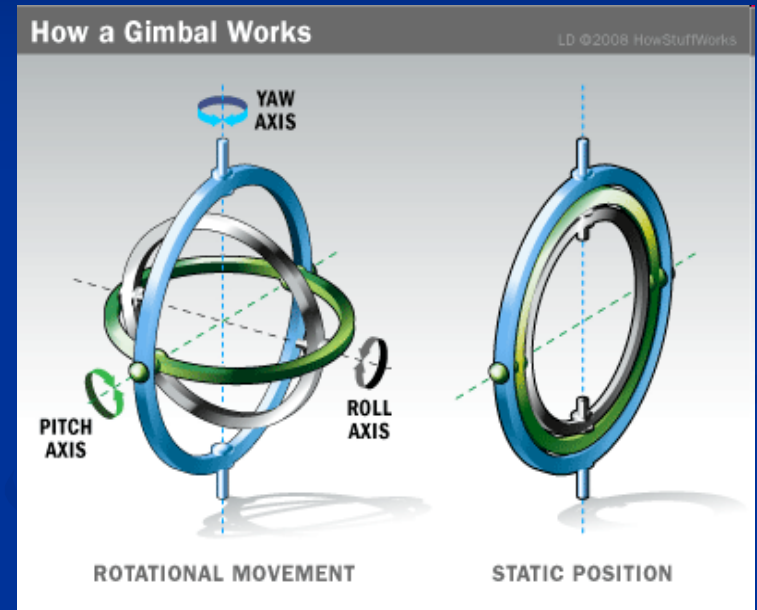
```
glRotatef(angleX,1,0,0);  
glRotatef(angleY,0,1,0);  
glRotatef(angleZ,0,0,1);  
drawObject();
```



Gyroscope  
(From Wikipedia)

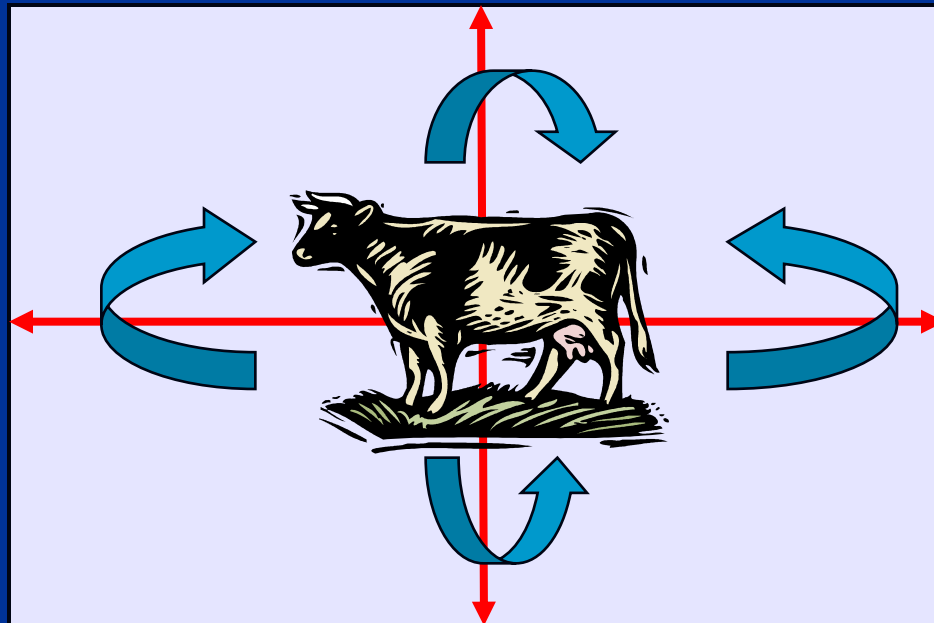
# Euler Angles

- Problem : Gimbal Lock
- Occurs when two axes are aligned
- Second and third rotations have effect of transforming earlier rotations
  - ex: Rot x, Rot y, Rot z
    - If  $\text{Rot } y = 90 \text{ degrees}$ ,  
 $\text{Rot } z == -\text{Rot } x$



# Arcball Interface

- Intuition : Make use of the mouse position to control object orientation
  - Rotate object about some axis based on mouse movement



# Arcball Interface

- Keep track a global rotation matrix  $\mathbf{R_g}$ .
- Whenever there is a mouse movement, create a new rotation  $\mathbf{R_n}$ .
- Update global rotation matrix  $\mathbf{R_g} = \mathbf{R_n} * \mathbf{R_g}$ .
- How to define  $\mathbf{R_n}$  ?



# Arcball Interface

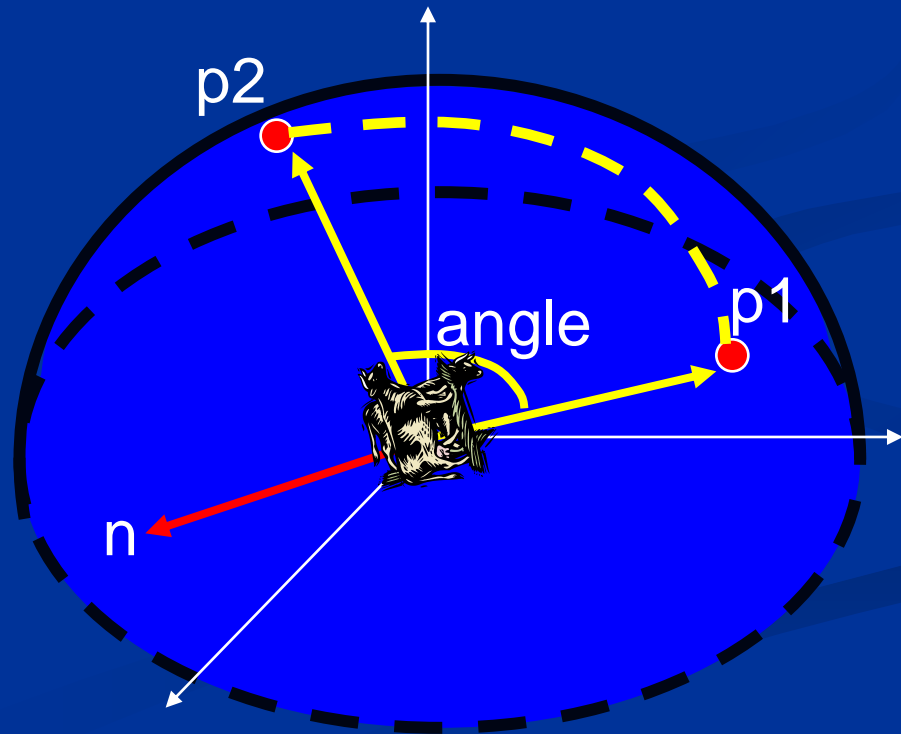
- To define a rotation : axis & angle
- Think of orientation as a point on the unit hemi-sphere
- How to rotate p1 to p2 ?

$$n = p1 \times p2$$

$$\text{axis} = n / |n|$$

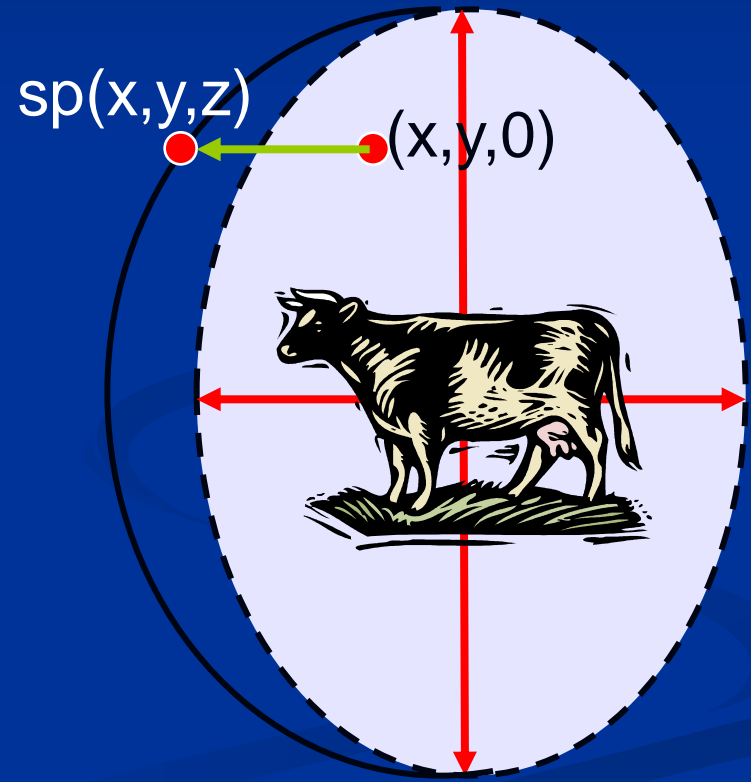
$$|n| = \sin(\text{angle})$$

$$\text{angle} = \arcsin(|n|)$$



# Arcball Interface

- How to find a point on sphere based on normalized screen coordinates ?
- Map a 2D point  $(x,y)$  back to a unit sphere
  - $z = \sqrt{1 - x^2 - y^2}$



# Arcball Interface

## ■ Summary:

- Get start/end mouse 2D position ( glutMotion )
- Map them to 3D points  $v_1, v_2$  on hemi-sphere
- Find rotation axis/angles from  $v_1, v_2$
- Update object orientation with rotation axis/angle
  - (Pre-multiply new rotation with current rotation)

# Rotation About Any Axis

- Check lecture note :

Let's suppose we have a unit direction vector

$$\mathbf{u} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \text{ where } x^2 + y^2 + z^2 = 1$$

We can derive a rotation by a given angle about this axis

$$\mathbf{R}(\theta, \mathbf{u}) = \mathbf{u}\mathbf{u}^T + \cos \theta (\mathbf{I} - \mathbf{u}\mathbf{u}^T) + (\sin \theta) \mathbf{u}^*$$

- You can also call `glRotate3f` to generate it.

# Rendering Acceleration

- Calling glBegin/glEnd is not optimal.
  - Many function calls
  - Repeated vertices
  - Data transfer
- Acceleration :
  - DrawArray
  - Display List
  - Vertex Buffer Object ( VBO )

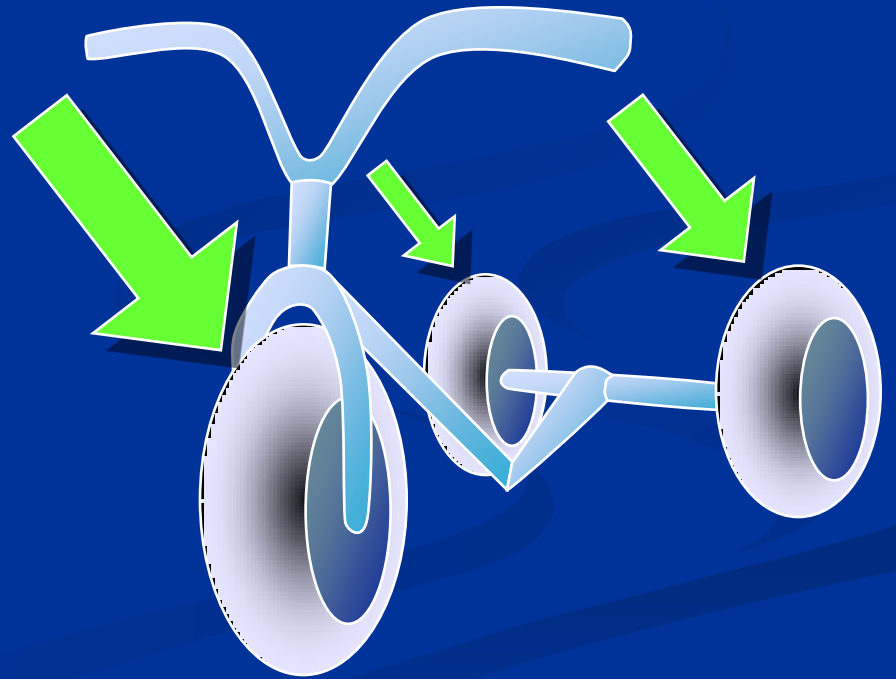
# Display Lists

Method One

# Display Lists

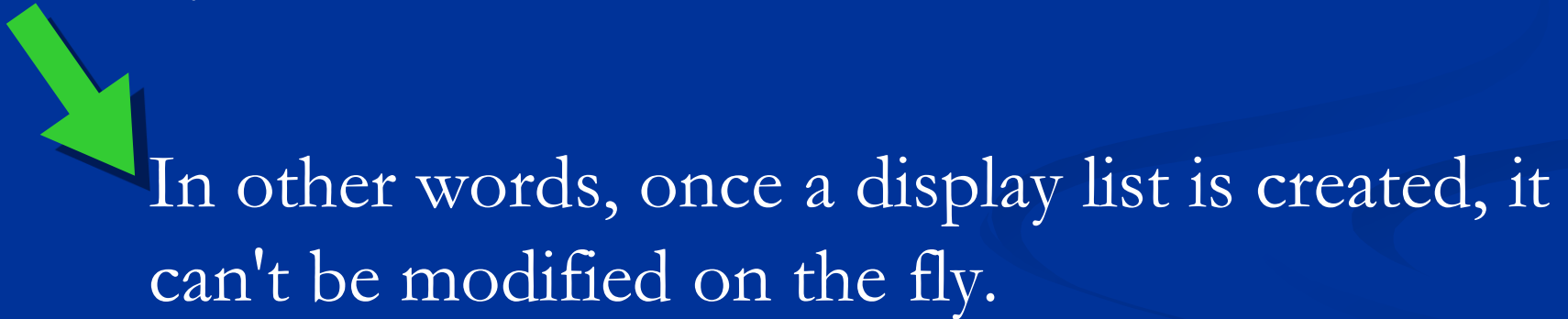
A display list is a convenient and efficient way to name and organize a set of OpenGL commands.

```
glCallList( wheel_id );  
modelview transformation  
glCallList( wheel_id );  
modelview transformation  
glCallList( wheel_id );
```



# Display Lists

To optimize performance, an OpenGL display list is a cache of commands rather than a dynamic database.



In other words, once a display list is created, it can't be modified on the fly.



# Display List

- A Display List is simply a group of OpenGL commands and arguments
- Most OpenGL drivers compile and accelerate Display Lists by
  - storing all static data on video ram
  - optimizing OpenGL commands execution
  - Frustum & occlusion culling
- Small driver overhead
- No time expensive data transfer

# Display List

- Usage : Create a new list
  - Call glBegin/glEnd / glVertex to store commands in the display list.
  - glCallList to reuse a display list.

glGenList  
glNewList  
glEndList  
glCallList  
.....

Red Book Sixth Edition :  
Chapter 7.

# Vertex Arrays

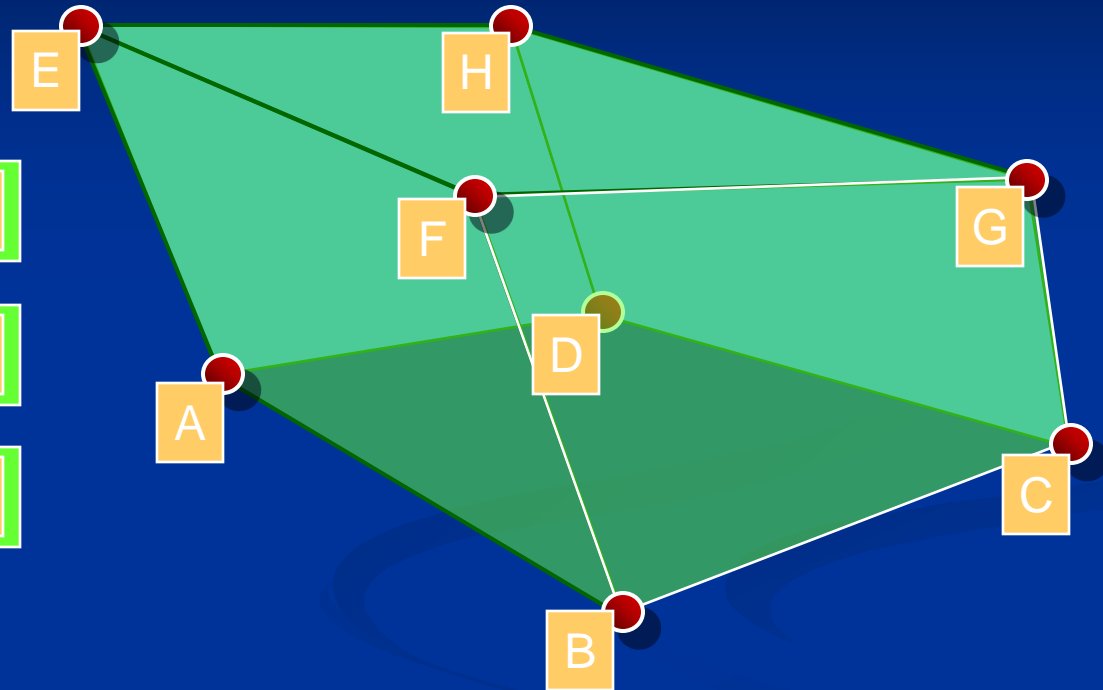
Method Two

# The Basic Idea

Vertices Stored in an Array

0	0	0	0	0	1	1	0
1	1	0	0	0	1	0	0
1	1	1	1	1	1	1	0

Vertex G



Indices of Quads into the vertex array

A	B	C	D	A	B	F	E	B	C	G	F	E	F	G	H
				A	D	H	E	D	C	G	H				

# Vertex Arrays

- Similar to conventional approach, but: One driver call for all vertices
  - small driver overhead
- Data resides in CPU memory.
  - Easier to update
- Still transferring all vertices
  - lot of transfer (CPU/AGP-bound bottleneck)

# Vertex Arrays

- Usage : Enable client state for vertex array.
  - Provide pointers to your vertices/faces in memory.
  - Call `glDrawElement` to rendering everything at once.

`glEnableClientState`

`glVertexPointer`

`glColorPointer`

`glDrawElements`

.....

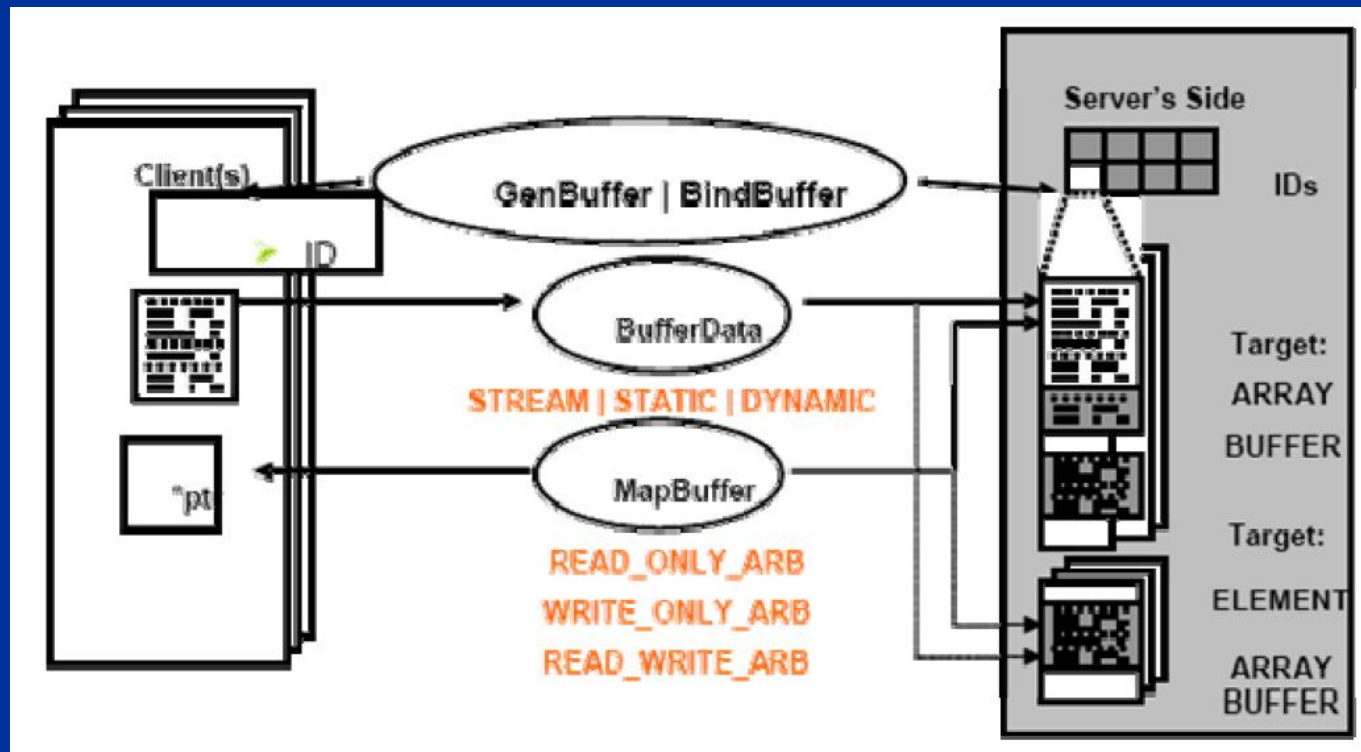
Red Book Sixth Edition :  
P. 65-81

# Buffer Object

Method Three

# Vertex Buffer Object (VBO)

- A vertex buffer object (VBO) is a powerful feature that allows storing vertex data in video ram





# Vertex Buffer Object (VBO)

- Very similar to vertex arrays
- VBOs hold geometry and state on the graphics hardware
  - Significant reduction in rendering time
- Provide mapping from application memory to graphics memory
  - Allows fast updates when geometry changes

# Vertex Buffer Object

- Usage : Allocate enough buffer space in video memory.
  - Maps buffer memory to represent vertex/indices data.
  - Render as vertex arrays.

glGenBuffers  
glBindBuffers  
glBufferData  
.....

Red Book Sixth Edition :  
P. 82-91

# Summary

- Use Display Lists or Vertex Buffer Objects to store **static objects**
- Vertex Arrays or dynamic Vertex Buffer for **deformable objects**
- DrawElements is expensive
  - draw as many Triangles per DrawElements as possible
- Keep data transfer as small as possible

# Q&A