

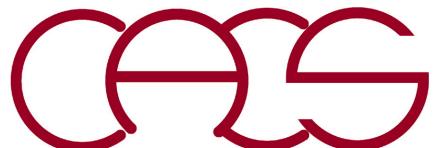
Scientific Visualization Basics

Aiichiro Nakano

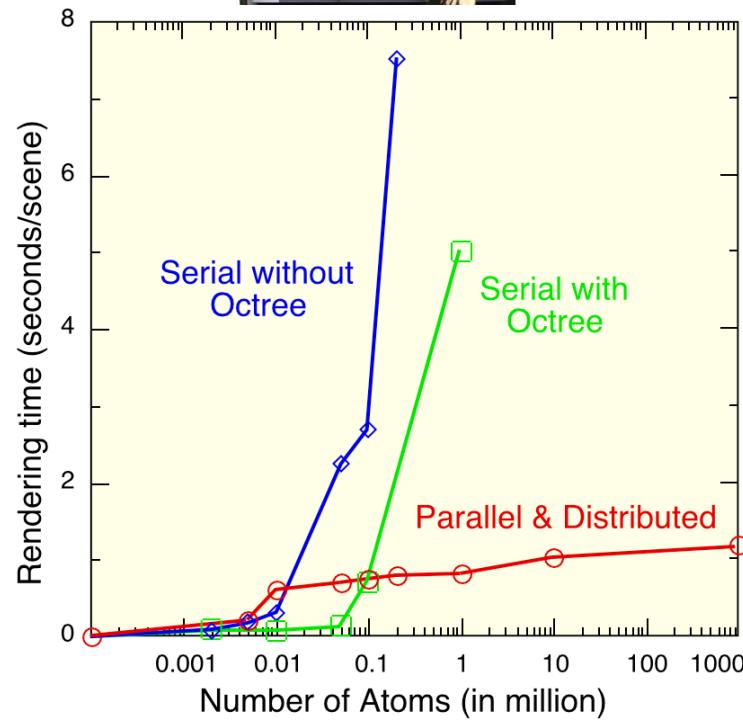
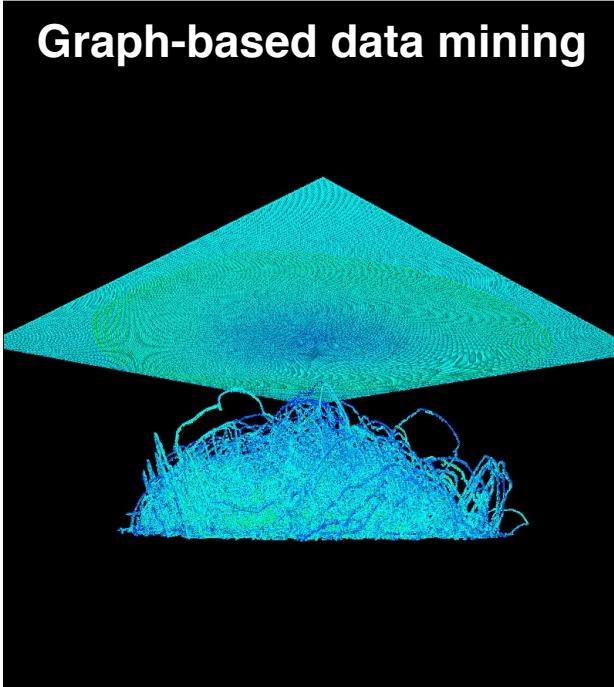
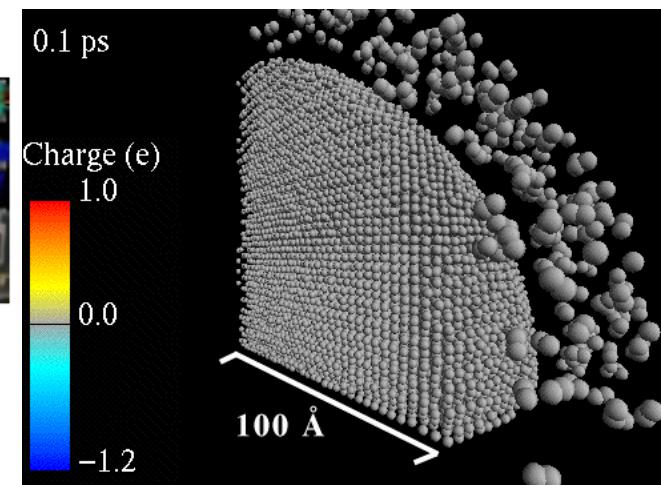
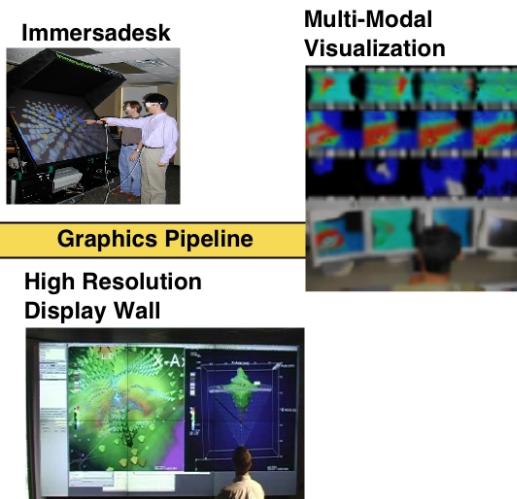
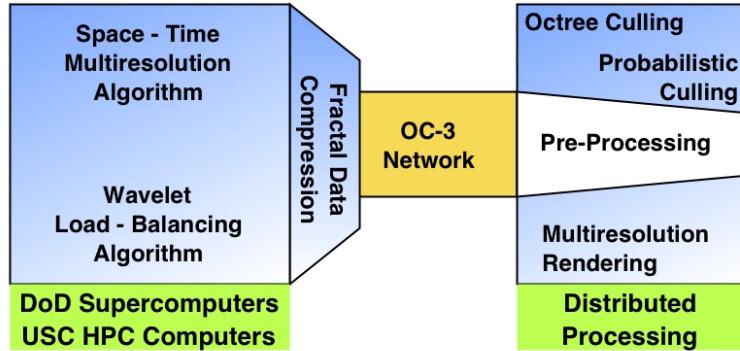
*Collaboratory for Advanced Computing & Simulations
Dept. of Computer Science, Dept. of Physics & Astronomy,
Dept. of Chemical Engineering & Materials Science,
Department of Biological Sciences
University of Southern California*

Email: anakano@usc.edu

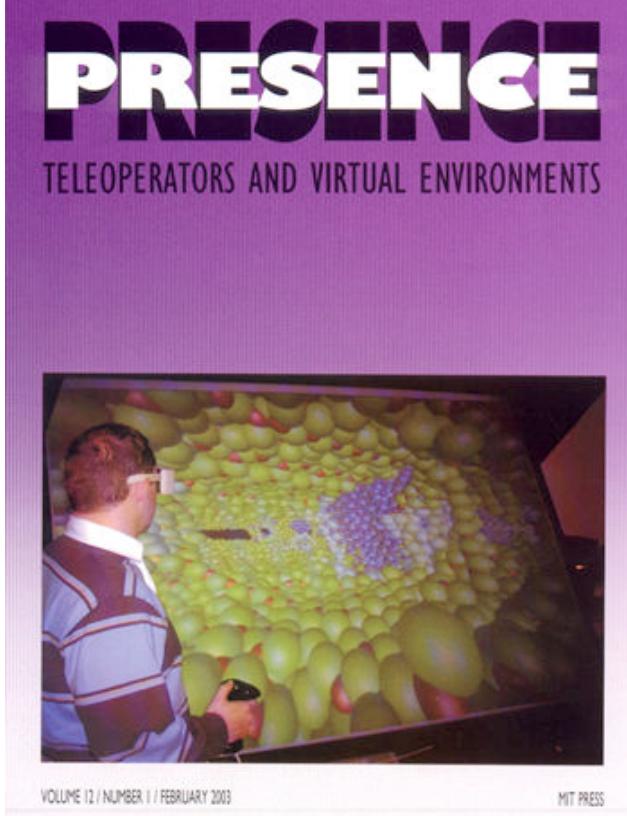
Goal: Experience simple OpenGL visualization of real simulation



Massive Scientific Data Visualization



Interactive visualization of billion atoms



OpenGL: Getting Started

Installing OpenGL & GLUT libraries:

- OpenGL: Standard, hardware-independent interface to graphics hardware.
- GLUT (OpenGL Utility Toolkit): Window-system-independent toolkit for window APIs.

<http://www.opengl.org>

Do it on your laptop!

<http://web.eecs.umich.edu/~sugih/courses/eecs487/glut-howto>

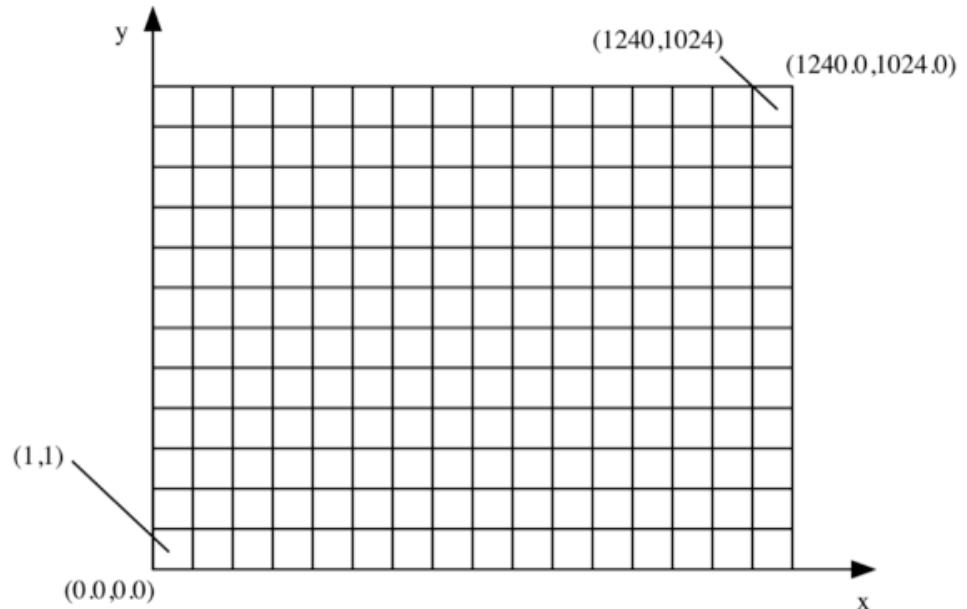
OpenGL Programming Basics

```
#include <OpenGL/gl.h>      // Header File For The OpenGL32 Library
#include <OpenGL/glu.h>      // Header File For The GLu32 Library
#include <GLUT/glut.h>        // Header File For The GLut Library

glutInit(&argc, argv);

/* Set up an window */
glutInitDisplayMode(GLUT_DOUBLE|GLUT_RGBA|GLUT_DEPTH); /*Initialize display mode*/
glutInitWindowSize(winx, winy); /* Specify window size */
glutCreateWindow("Lennard-Jones Atoms"); /* Open window */
 glEnable(GL_DEPTH_TEST); /* Enable z-buffer for hidden surface removal */
 glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT); /* Clear the window */
```

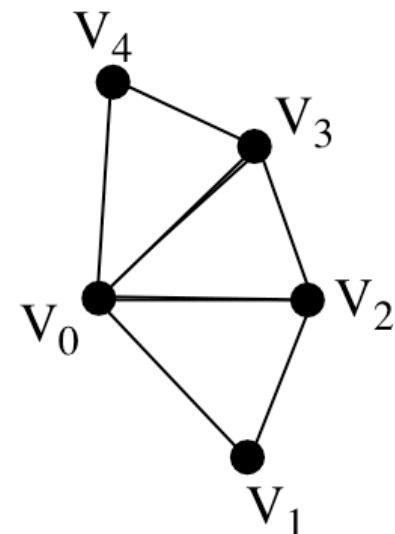
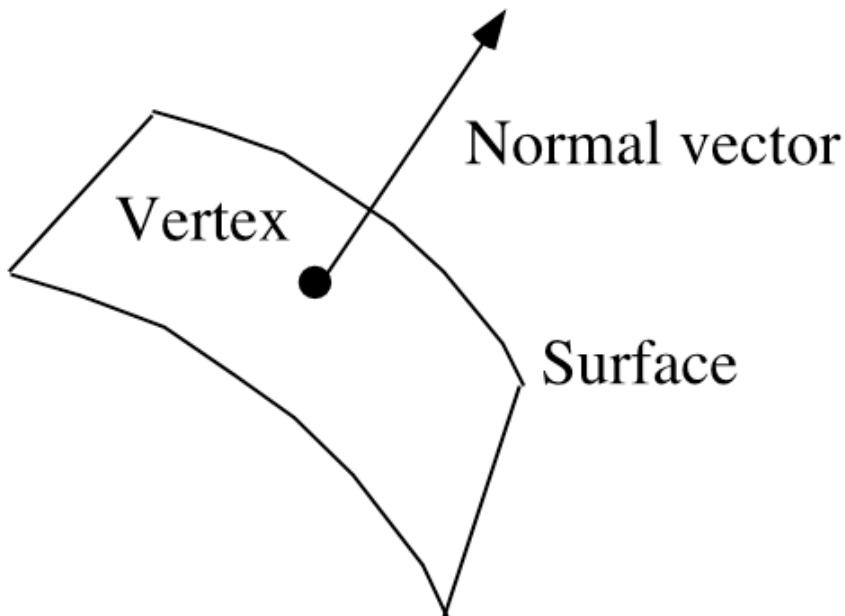
- **Frame buffer:** A collection of buffers in memory, which store data for screen pixels (e.g., 1280 pixels wide & 1024 pixels high) such as color, depth information for hidden surface removal, etc.



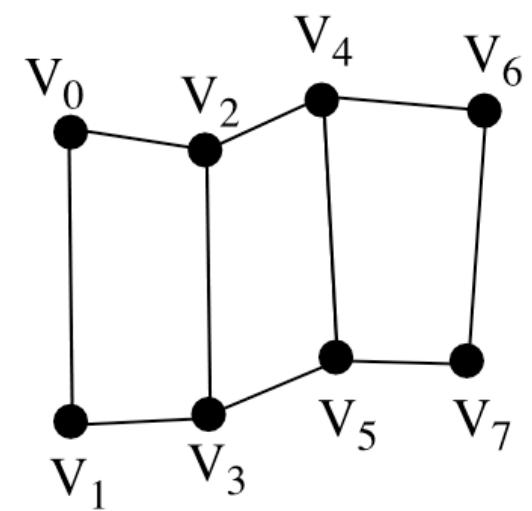
OpenGL Event-Handling Loop

Polygonal Surfaces

```
float normal_vector[MAX_VERTICES][3],vertex_position[MAX_VERTICES][3];
glBegin(GL_QUAD_STRIP);
    for (i=0; i<number_of_vertices; i++) {
        glNormal3f(normal_vector[i]);
        glVertex3f(vertex_position[i]);
    }
glEnd();
```



GL_TRIANGLE_FAN



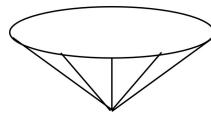
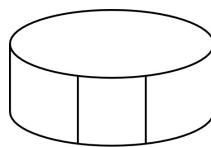
GL_QUAD_STRIP

Polygonal Sphere

```

int nlon=3, nlat=2;
loninc = 2*M_PI/nlon; /* Δφ */
latinc = M_PI/nlat; /* Δθ */
/* South-pole triangular fan */
glBegin(GL_TRIANGLE_FAN);
    glNormal3f(0,-1,0);
    glVertex3f(0,-radius,0);
    lon = 0;
    lat = -M_PI/2 + latinc;
    y = sin(lat);
    for (i=0; i<=nlon; i++) {
        x = cos(lon)*cos(lat);
        z = -sin(lon)*cos(lat);
        glNormal3f(x,y,z);
        glVertex3f(x*radius,y*radius,z*radius);
        lon += loninc;}
    glEnd();
/* Quadrilateral strips to cover the sphere */
for (j=1; j<nlat-1; j++) {
    lon = 0;
    glBegin(GL_QUAD_STRIP);
    for (i=0; i<=nlon; i++) {
        x = cos(lon)*cos(lat);
        y = sin(lat);
        z = -sin(lon)*cos(lat);
        glNormal3f(x,y,z);
        glVertex3f(x*radius,y*radius,z*radius);
        x = cos(lon)*cos(lat+latinc);
        y = sin(lat+latinc);
        z = -sin(lon)*cos(lat+latinc);
        glNormal3f(x,y,z);
        glVertex3f(x*radius,y*radius,z*radius);
        lon += loninc;}
    glEnd();
    lat += latinc;}

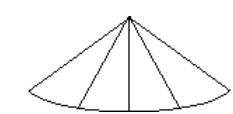
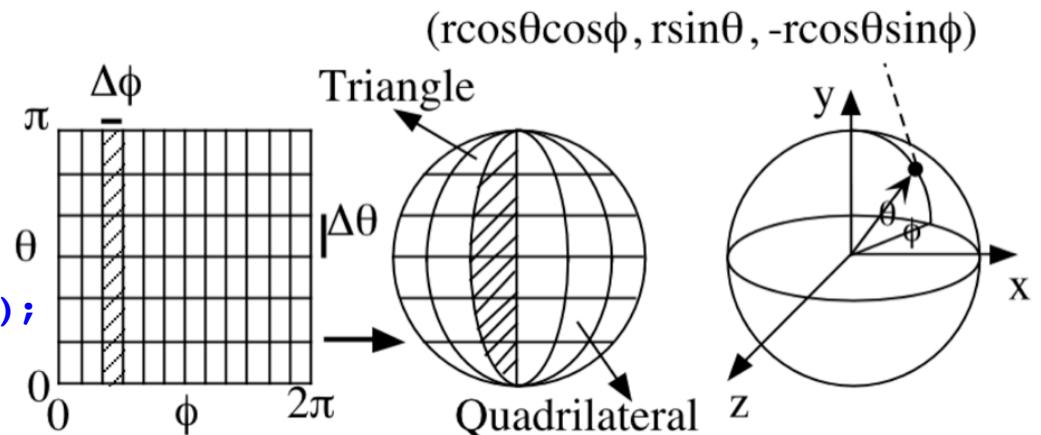
```

```

/* North-pole triangular fan */
glBegin(GL_TRIANGLE_FAN);
    glNormal3f(0,1,0);
    glVertex3f(0,radius,0);
    y = sin(lat);
    lon = 0;
    for (i=0; i<=nlon; i++) {
        x = cos(lon)*cos(lat);
        z = -sin(lon)*cos(lat);
        glNormal3f(x,y,z);
        glVertex3f(x*radius,y*radius,z*radius);
        lon += loninc;
    }
    glEnd();

```

Display Lists

- **Display list:** A group of OpenGL commands that have been stored for later execution.

```
/* Generates one new display-list ID */
GLuint sphereid = glGenLists(1);

/* Define a routine to draw a sphere*/
glNewList(sphereid, GL_COMPILE);
    ...code to draw a sphere...
glEndList();

/* Execute sphere drawing */
glCallList(sphereid);
```

Transformation Matrix

- **Transformation matrix:** Specifies the amount by which the object's coordinate system is to be rotated, scaled, or translated, *i.e.*, affine transformation.
- **Matrix stack:** A stack of transformation matrices—at the top of the stack is the current transformation matrix. Initially the transformation matrix is the identity matrix.

$$\begin{aligned} \dot{r}' &= \ddot{A}\dot{r} + \dot{b} \\ \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} &= \begin{bmatrix} a_{11} & a_{12} & a_{13} & b_1 \\ a_{21} & a_{22} & a_{23} & b_2 \\ a_{31} & a_{32} & a_{33} & b_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \end{aligned}$$

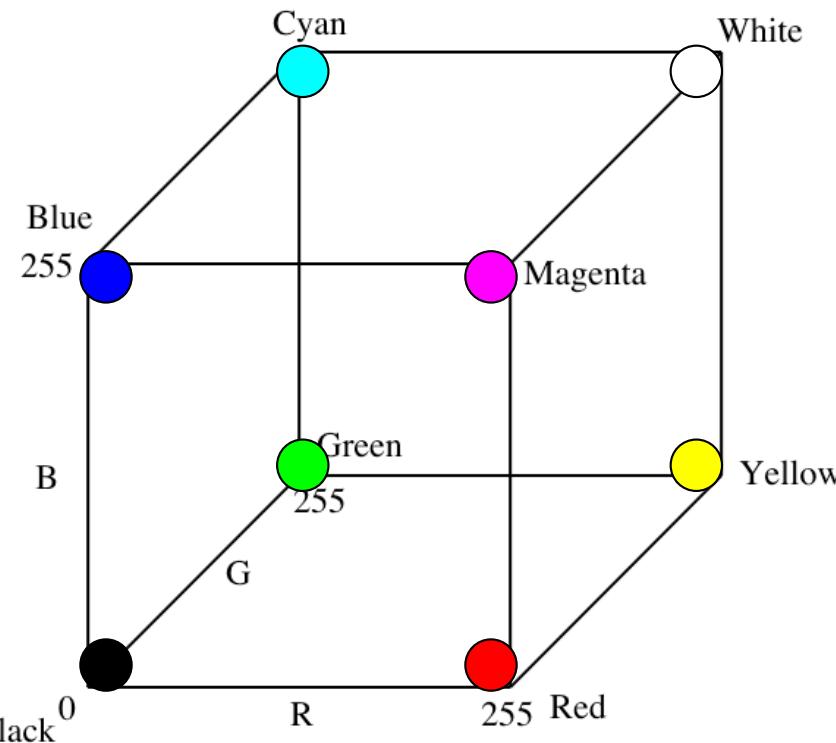
Matrix Identity
= {1, 0, 0, 0,
0, 1, 0, 0,
0, 0, 1, 0,
0, 0, 0, 1};

```
glPushMatrix();
glTranslatef(atoms[i].crd[0],atoms[i].crd[1],atoms[i].crd[2]);
glCallList(sphereid);
glPopMatrix();
```

Color Display

- **RGB(A) mode:** Specifying color by providing red, green & blue intensities (& alpha component).
- **Alpha component:** Specifies the opacity of a material; default value is 1.0 (nontransparent), if not specified.

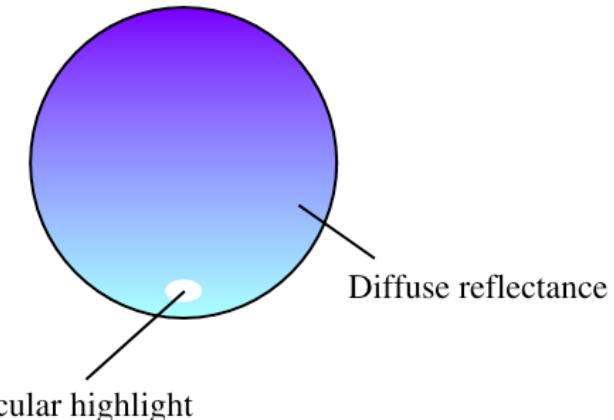
```
float r=1.0; g=0.0; b=0.0;  
  
glColor3f(r,g,b);
```



- **OpenGL as a state machine:** Color change stays.

Lighting & Materials

OpenGL color =
light \times material-reflectance



OpenGL Color Types

- **Diffuse component:** Gives the appearance of a matter or flat reflection from an object's surface.
- **Ambient illumination:** Simulates light reflected from other objects.
- **Specular light:** Creates highlights.
- **Emission:** Simulates the appearance of lights in the scene.

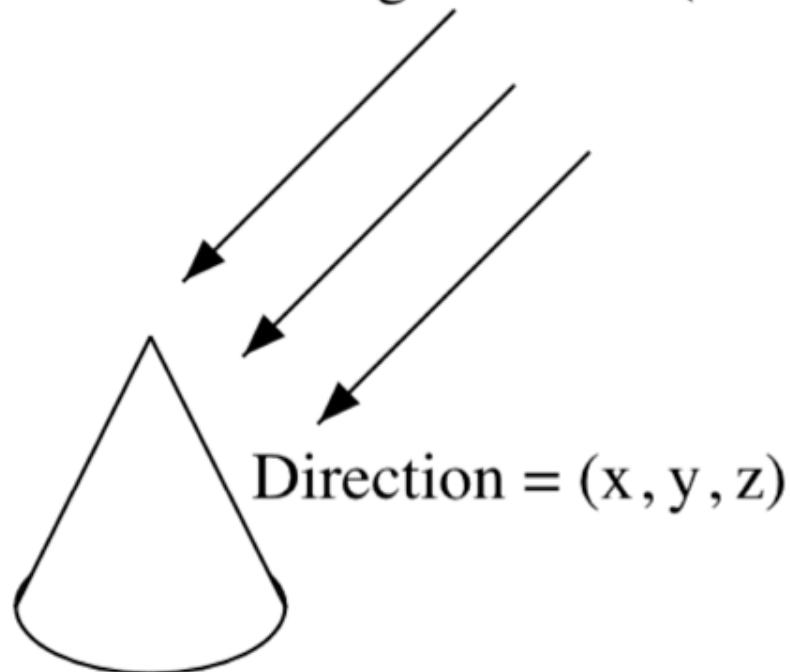
Materials Definition

- **Relectance:** Material is characterized by ambient, diffuse & specular reflectance, i.e., how the object reflects light.
- **glEnable(GL_COLOR_MATERIAL)**
In this mode, the current color specified by **glColor*** () will change the ambient & diffuse reflectance.

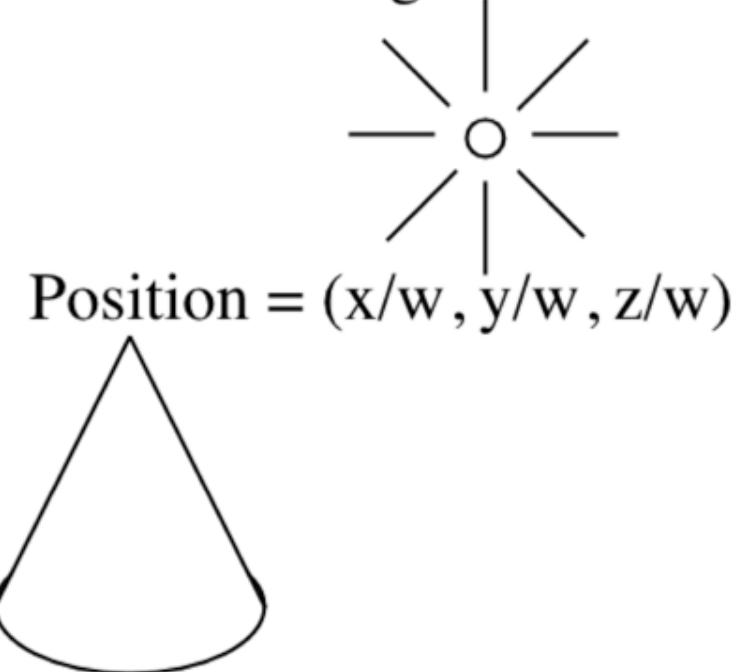
Lighting Source

```
float light_diffuse[4] = {1.0,1.0,1.0,1.0};  
float light_position[4] = {0.5,0.5,1.0,0.0};  
  
/* Define a lighting source */  
glLightfv(GL_LIGHT0,GL_DIFFUSE,light_diffuse);  
glLightfv(GL_LIGHT0,GL_POSITION,light_position);  
  
/* Enable a single OpenGL light */  
 glEnable(GL_LIGHTING);  
 glEnable(GL_LIGHT0);
```

Directional light source ($w = 0$)



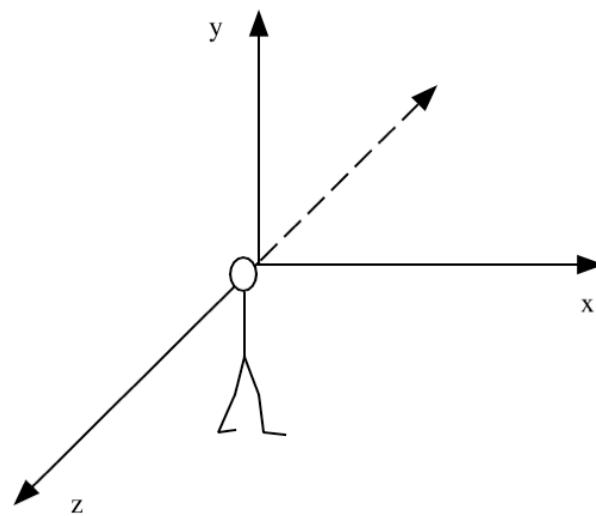
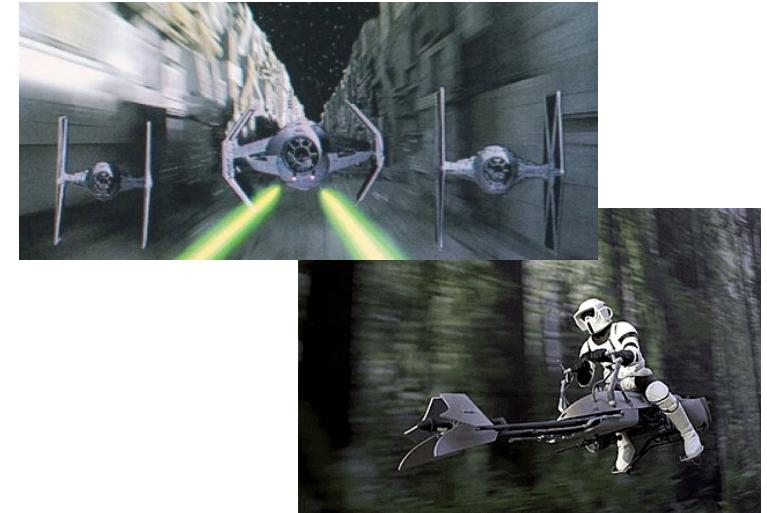
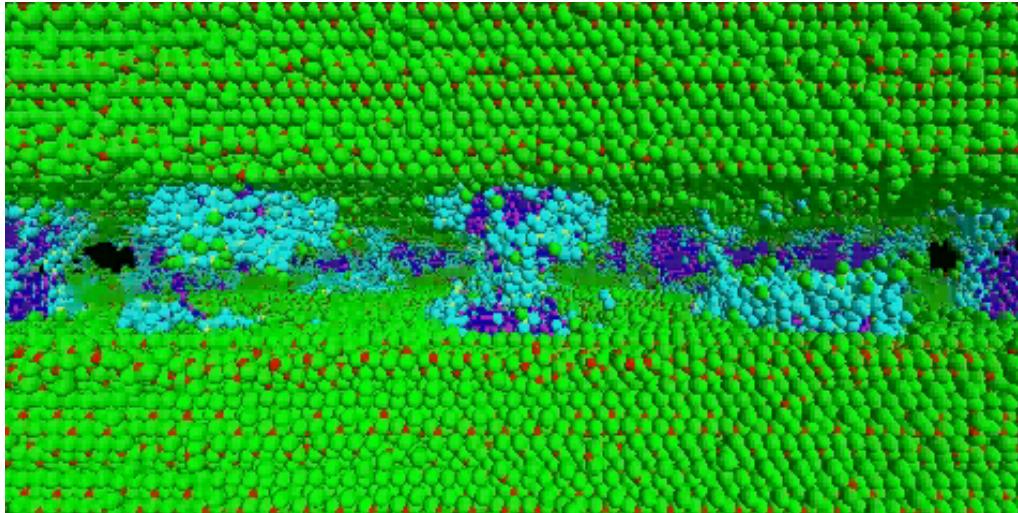
Point light source



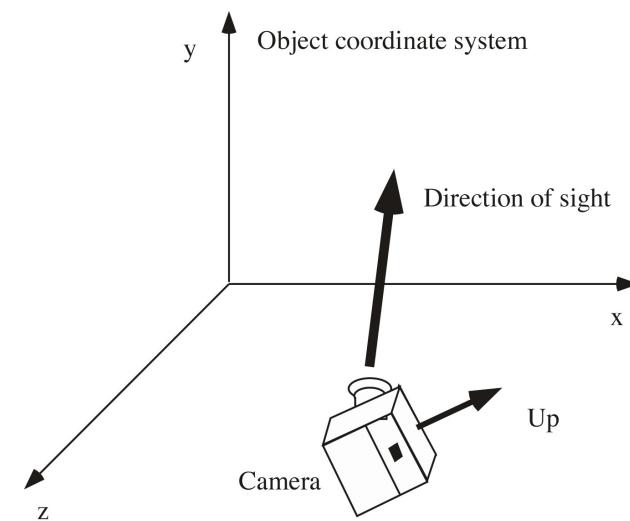
Viewing Transformation

- Viewing transformation: Transforms object coordinates to eye coordinates.

```
gluLookat(eyx, eyey, eyz, centerx, centery, centerz, upx, upy, upz);
```



Eye coordinate system



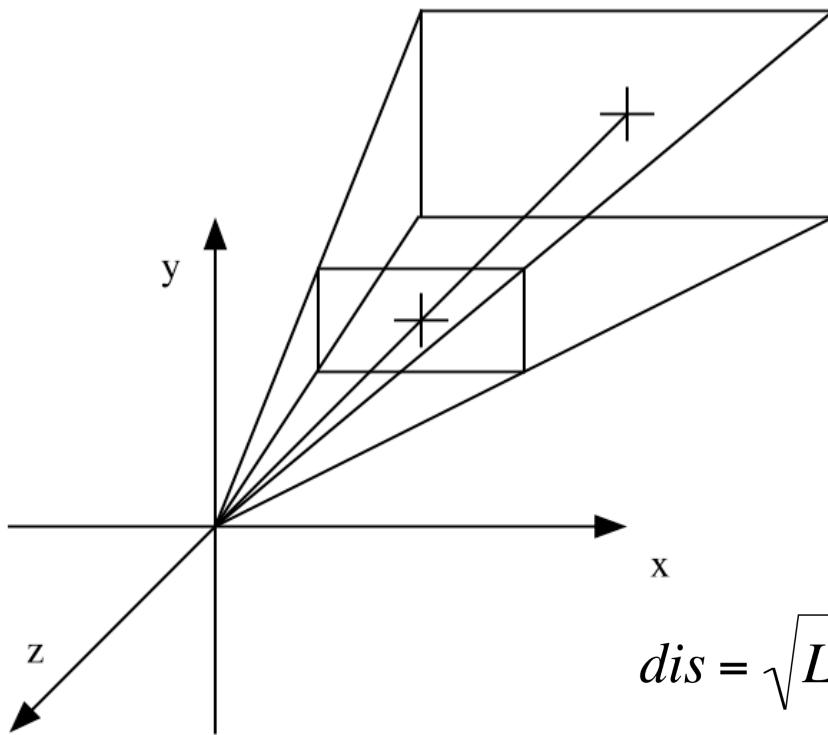
Camera specification

Clipping

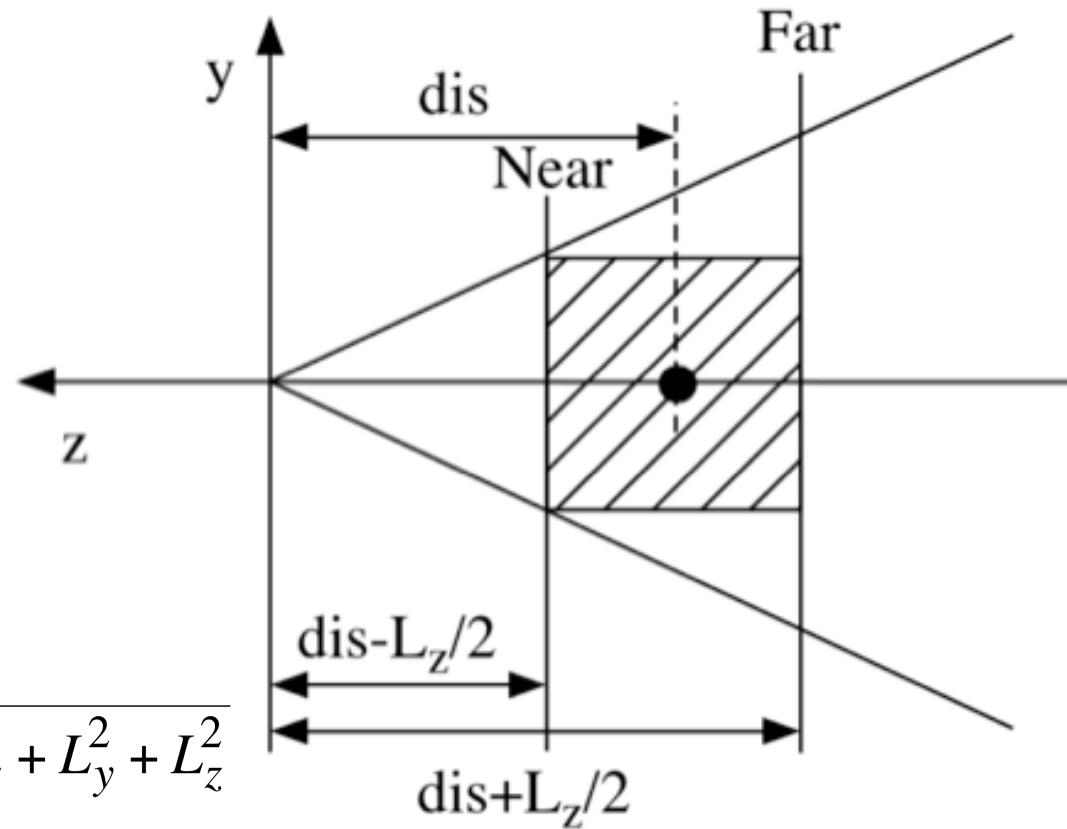
```
void reshape (int w, int h) {  
    ...  
    /* set the GL viewport to match the full size of the window */  
    glViewport(0, 0, (GLsizei)w, (GLsizei)h);  
    aspect = w/(float)h;  
    glMatrixMode(GL_PROJECTION);  
    glLoadIdentity();  
    gluPerspective(fovy,aspect,near_clip,far_clip);  
    glMatrixMode(GL_MODELVIEW);  
}
```

$$fovy = 2 \tan^{-1} \left(\frac{L_y/2}{dis - L_z/2} \right)$$

View Frustum

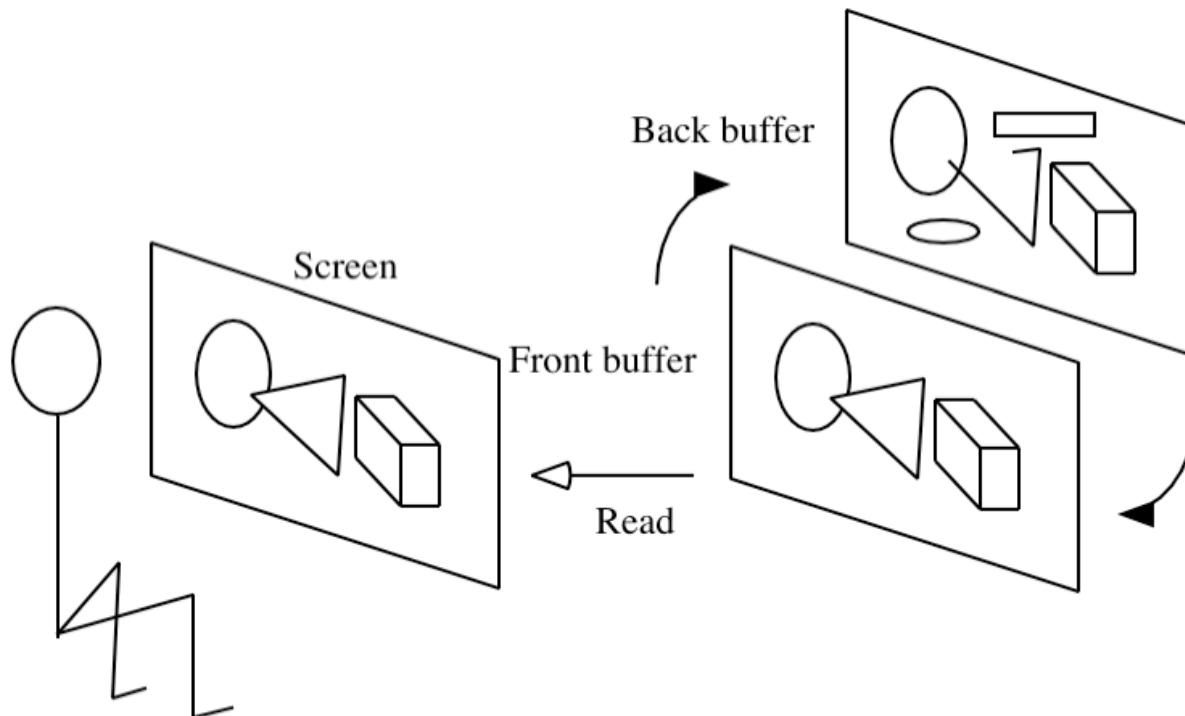


$$dis = \sqrt{L_x^2 + L_y^2 + L_z^2}$$



Animation

```
main() {
    glutIdleFunc/animate);
}
...
void animate() { /* Callback function for idle events */
/* Keep updating the scene until the last MD step is reached */
    if (stepCount <= StepLimit) {
        SingleStep(); /* One MD-step integration */
        if (stepCount%StepAvg == 0) EvalProps();
        makeCurframeGeom(); /* Redraw the scene */
        glutPostRedisplay();
        ++stepCount;
    }
}
```



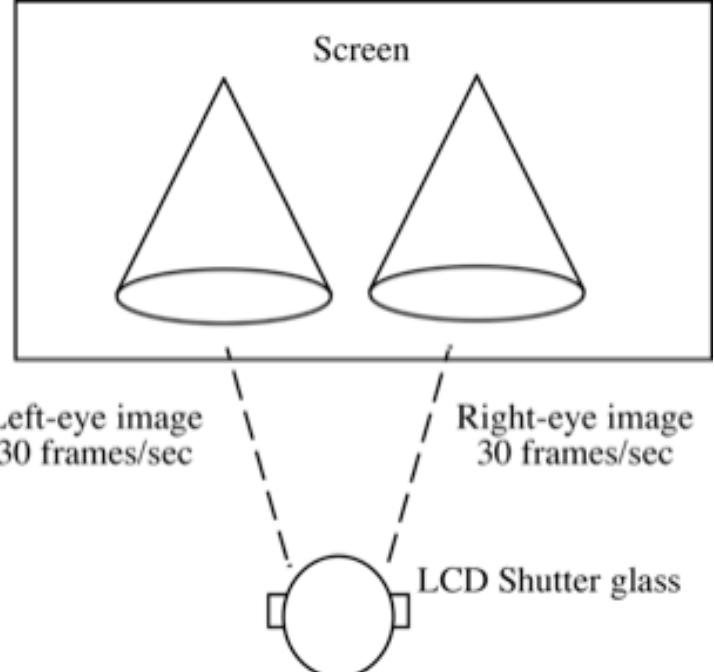
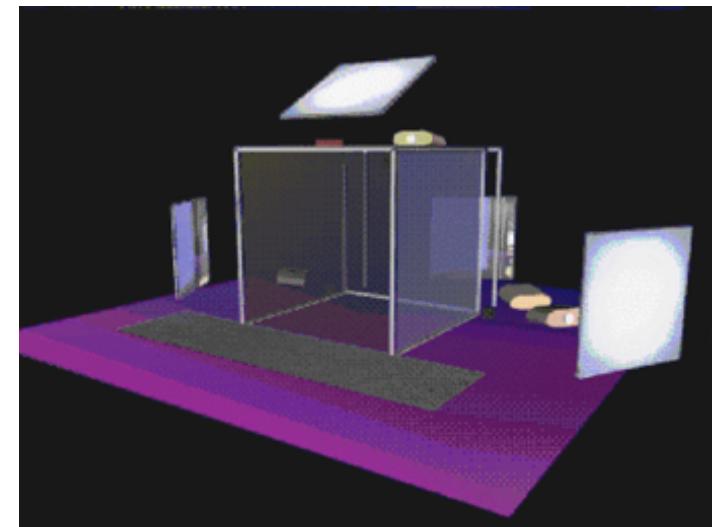
```
void display() {
    ...
    drawScene();
    glutSwapBuffers();
}
```

Immersive & Interactive Visualization



ImmersaDesk
at CACS

CAVE



- **Stereographics**
- **Tracking system**
- **Wand: 3D (6 degrees-of-freedom)
mouse**

Origin: Sutherland (1968)

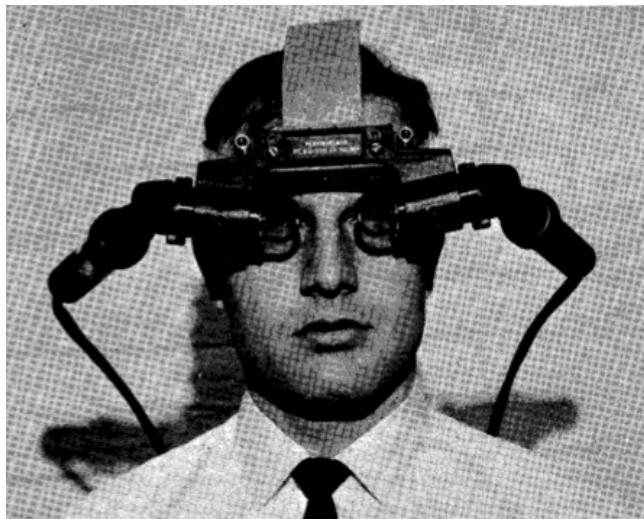


FIGURE 2—The head-mounted display optics with miniature CRT's



FIGURE 4—The ultrasonic head position sensor in use

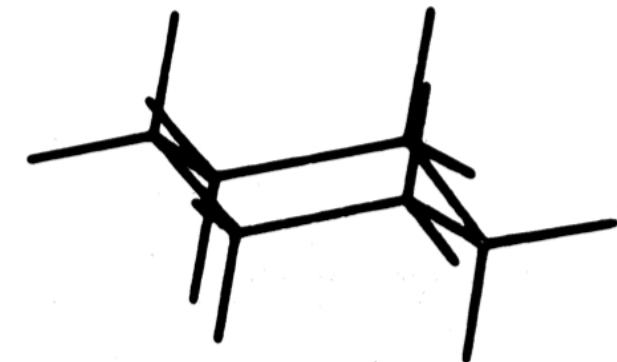


FIGURE 8—A computer-displayed perspective view of the cyclo-hexane molecule

Scientific VR

Journal of Molecular Graphics and Modelling 65 (2016) 94–99



Contents lists available at ScienceDirect

Journal of Molecular Graphics and Modelling

journal homepage: www.elsevier.com/locate/JMGM



iBET: Immersive visualization of biological electron-transfer dynamics



C. Masato Nakano ^a, Erick Moen ^b, Hye Suk Byun ^c, Heng Ma ^d, Bradley Newman ^e,
Alexander McDowell ^e, Tao Wei ^{d,*}, Mohamed Y. El-Naggar ^{c,f,g,**}

SoftwareX 9 (2019) 112–116



Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx



Original software publication

Game-Engine-Assisted Research platform for Scientific computing (GEARS) in Virtual Reality



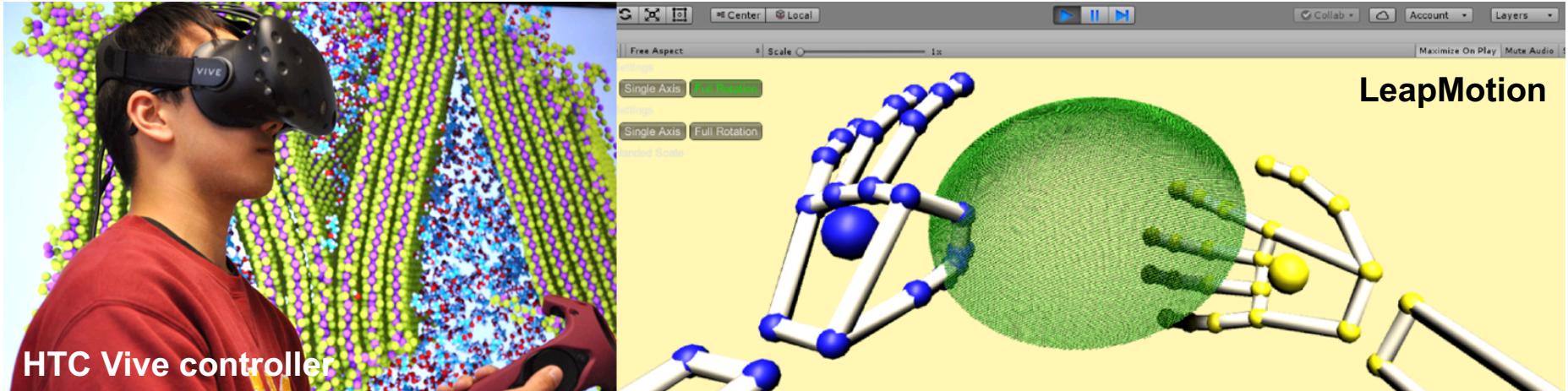
Brandon K. Horton ^a, Rajiv K. Kalia ^{a,b,c,d}, Erick Moen ^{b,e}, Aiichiro Nakano ^{a,b,c,d,f},
Ken-ichi Nomura ^{a,d,*}, Michael Qian ^a, Priya Vashishta ^{a,b,c,d}, Anders Hafreager ^g

<https://github.com/USCCACS/GEARS>

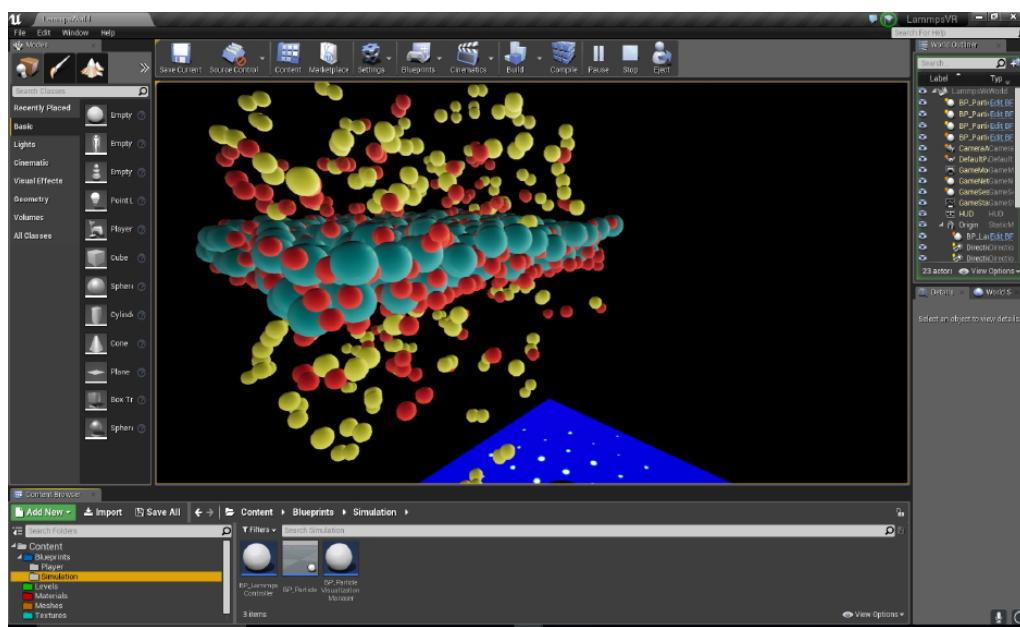


Scientific VR Use Cases

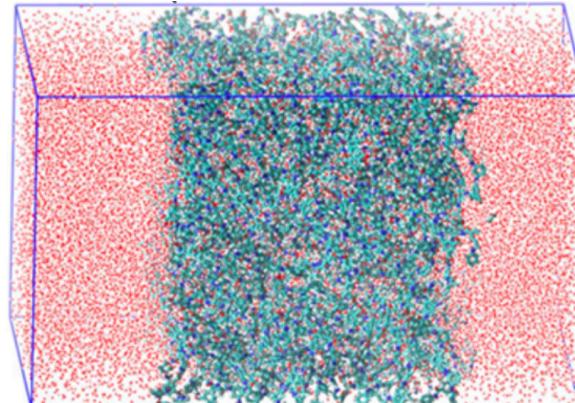
Interact with data



Dynamic linking to LAMMPS molecular-dynamics (MD) code



Virtual confocal microscopy



CAVE Library Programming

```
#include <cave_ogl.h>
#include <GL/glu.h>

GLUquadricObj *sphereObj;
void init_gl(void) {
    float redMaterial[] = { 1, 0, 0, 1 };
    glEnable(GL_LIGHT0);
    glMaterialfv(GL_FRONT_AND_BACK, GL_AMBIENT_AND_DIFFUSE, redMaterial);
    sphereObj = gluNewQuadric();
}

void draw_ball(void) {
    glClearColor(0., 0., 0., 0.);
    glClear(GL_DEPTH_BUFFER_BIT|GL_COLOR_BUFFER_BIT);
    glEnable(GL_LIGHTING);
    glPushMatrix();
    glTranslatef(0.0, 4.0, -4.0);
    gluSphere(sphereObj, 1.0, 8, 8);
    glPopMatrix();
    glDisable(GL_LIGHTING);
}
main(int argc,char **argv) {
    CAVEConfigure(&argc,argv,NULL); /* Initialize the CAVE */
    CAVEInit();
    CAVEInitApplication(init_gl,0); /* Pointer to the GL initialization function */
    CAVEDisplay(draw_ball,0); /* Pointer to the drawing function */
    while (!CAVEgetbutton(CAVE_ESCKEY)) /* Wait for the escape key to be hit */
        sginap(10); /* Nap so that this busy loop doesn't waste CPU time */
    CAVEExit(); /* Clean up & exit */
}
```

<http://cacs.usc.edu/education/cs653.html> → ball.c