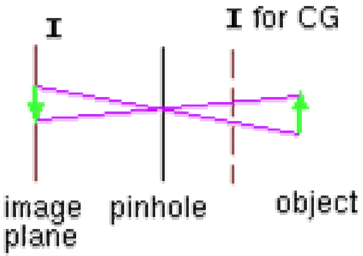
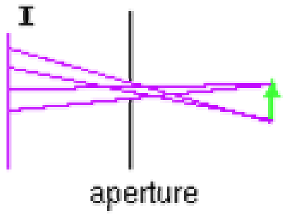
Cameras (and eye)

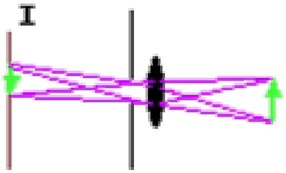
Ideal Pinhole



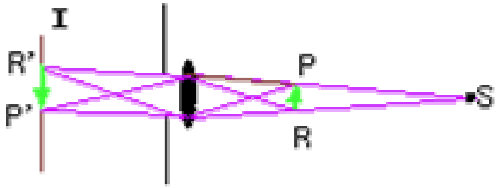
Real Pinhole



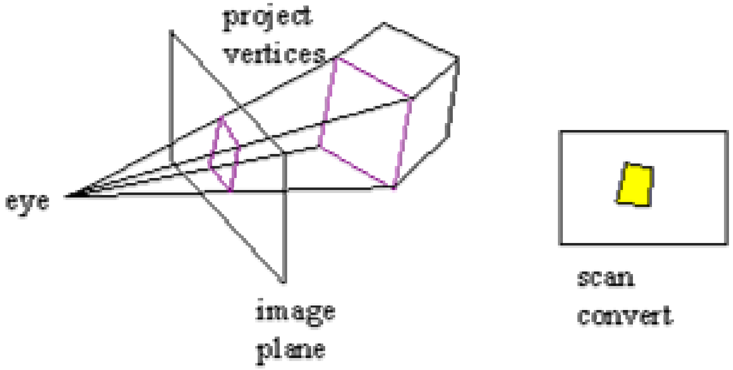
Real + lens



Depth of field



How do we draw objects?



Z-buffer

- Polygon Based
- Fast

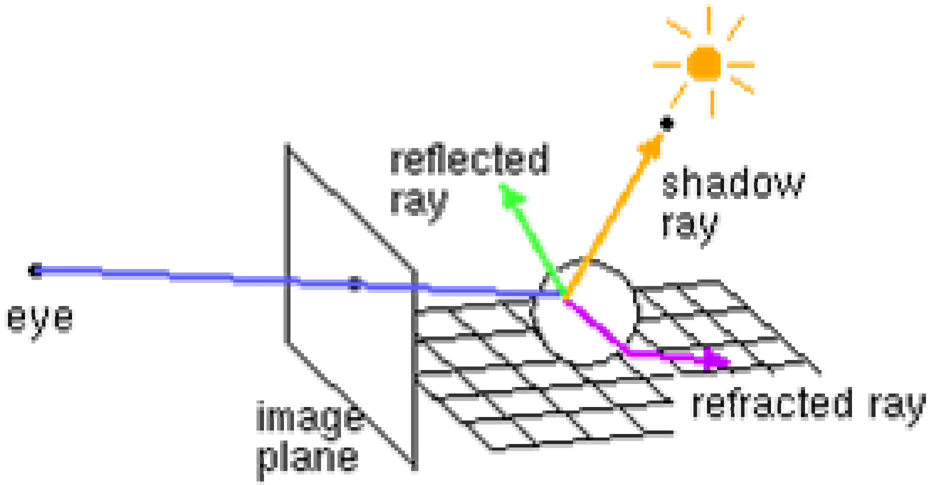
Raytracing

- Ray/Object intersections
- Slow

Copyright Pixar

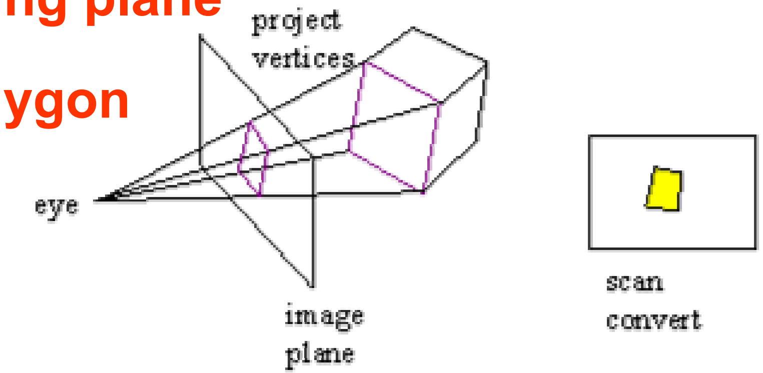
Raytracing

```
for each pixel on screen
  determine ray from eye through pixel
  find closest intersection of ray with an object
  cast off reflected and refracted ray, recursively
  calculate pixel colour, draw pixel
end
```

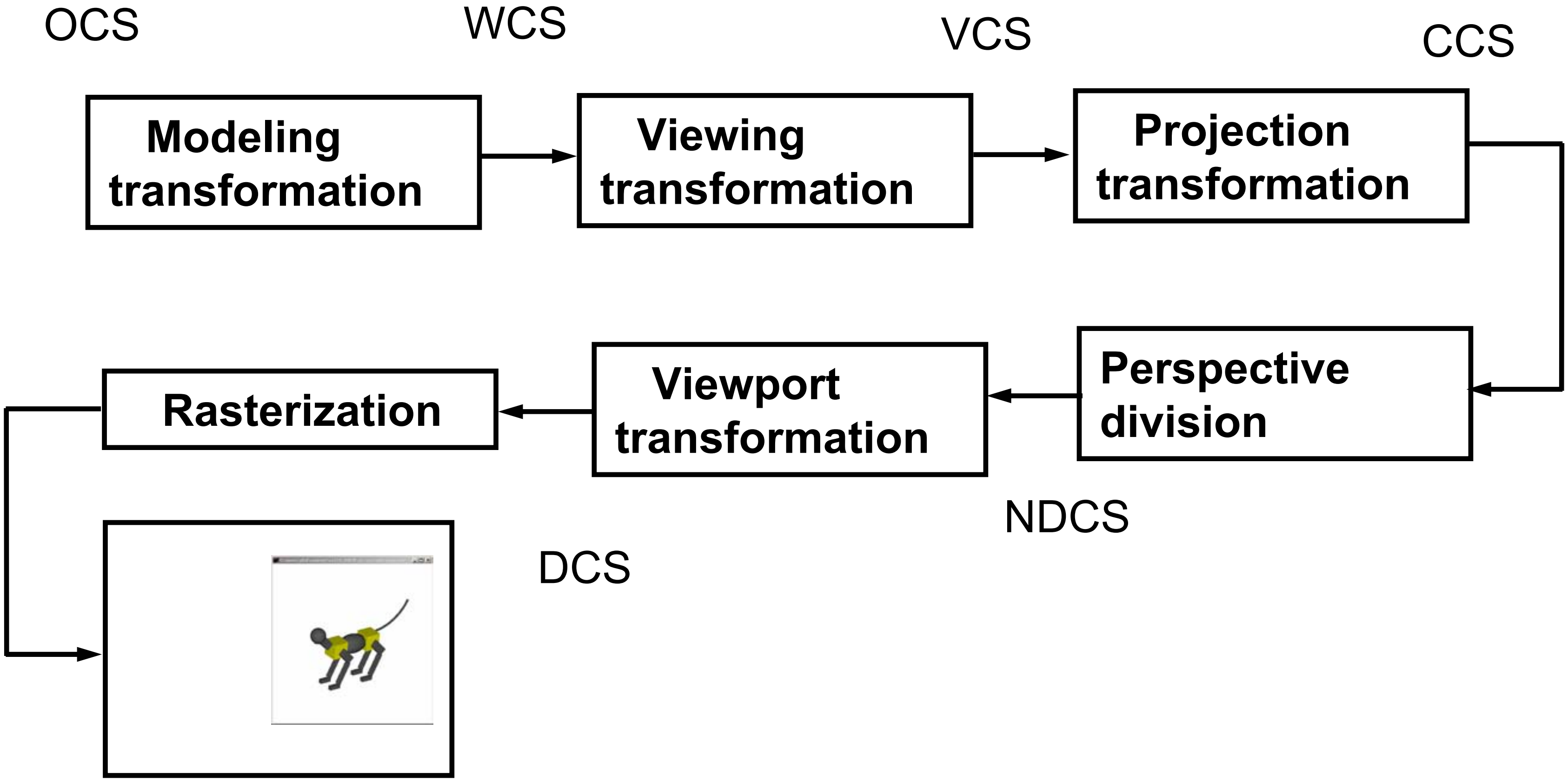


Z-buffer algorithm

```
for each polygon in model
  project vertices of polygon onto viewing plane
  for each pixel inside the projected polygon
    calculate pixel colour
    calculate pixel z-value
    compare pixel z-value to value stored for pixel in z-buffer
    if pixel is closer, draw it in frame-buffer and z-buffer
  end
end
```



Z-buffer Graphics Pipeline



OCS

Modeling transformation

WCS

Viewing transformation

VCS

Projection transformation

CCS

Perspective division

NDCS

Viewport transformation

Rasterization

DCS

OpenGL®

What is Open GL

Open Graphics Standard Specification for INTERACTIVE 3D Graphics

- Specification governed by a consortium of companies
- Implementation is up to the manufacturer of graphics boards

What is it for us?

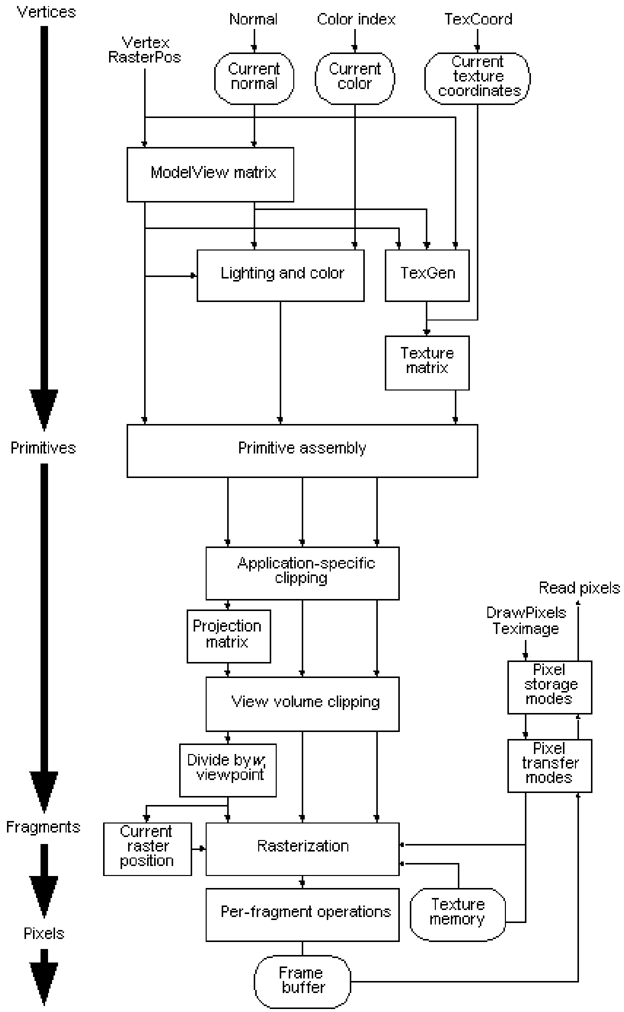
Open Graphics Standard

- API
- Library
- State Machine
- Pipeline

GL: Core

GLU: Higher level utilities

GLUT: Windowing and interaction



Example

Headers:

```
#include <GL/gl.h>
#include <GL/glu.h>
#include "GL/glut.h“
```

Libraries:

glut32.lib,

opengl32.lib,

glu32.lib

Dynamic libraries

glut32.dll

Setting up GLUT a Window

```
int main(int argc, char** argv)  {
  glutInitDisplayMode (GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
  glutInitWindowPosition (0, 0);
  glutInitWindowSize(800,800);
  glutCreateWindow(argv[0]);

  // register callbacks
  glutReshapeFunc (myReshapeCB);
  glutKeyboardFunc(myKeyboardCB );
  glutMouseFunc(myMouseCB) ;
  glutMotionFunc(myMotionCB) ;
  glutDisplayFunc(display);

  myinit() ;            // initialize
  glutMainLoop();       // start the main  loop
  return 0;             // never reached
}
```

Mouse callbacks

```
void myMouseCB(int button, int state, int x, int y) {    // start or end interaction
      if( button == GLUT_LEFT_BUTTON && state == GLUT_DOWN ) {
              printf("Left button down\n") ;
      }
      if( button == GLUT_LEFT_BUTTON && state == GLUT_UP ) {
              printf("Left button up\n") ;
      }
      glutPostRedisplay() ;          // Tell the system to redraw the window
}
void myMotionCB(int x, int y) {                // interaction (mouse motion)
      printf("Moving with button down\n") ;
      glutPostRedisplay() ;
}
```

Keyboard callback

```
void myKeyboardCB(unsigned char key, int x, int y) {
   switch (key) {
       case 'q':
       case 27:
               exit(0);
               break;
   }
}
```

Display function

```
void display(void)  {
      glMatrixMode(GL_PROJECTION) ;
      glLoadIdentity() ;
      glMatrixMode(GL_MODELVIEW) ;
      glLoadIdentity();

       glClearColor(1.0f,1.0f,1.0f,1.0f);   // set the background colour
      // OK, now clear the screen with the background colour
      glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
      glColor3f(0.5,0,0) ;              // set the current color
      drawSphere() ;                    // draw a sphere
      glutSwapBuffers();                // swap the buffers (show the image)
}
```

Elements of a scene in OpenGL

Geometric Primitives

Material properties

Light sources

Copyright Pixar

Primitives in OpenGL

Points

Lines

Curves (piece-wise linear approximation)

Polygons

Surfaces (polygonal approximation)

Types

GLint

GLfloat

GLdouble

Points

```
glBegin(GL_POINTS)
 glVertex3f(GLfloat x, GLfloat y, GLfloat z) ;
 glVertex2i(GLint x, GLint y) ;
 glVertex3dv(GLdouble p[3] ) ;
glEnd() ;
```

Point details

```
glPointSize(float size) ;
glColor3f(GLfloat r, GLfloat g, Glfloat b) ;
```

Lines

```
glBegin(GL_LINES)
 glVertex2i(x1,y1) ;
 glVertex2i(x2,y2) ;
 glVertex2i(x3,y3) ;
 glVertex2i(x4,y4) ;
glEnd()
```

Line strip

```
glBegin(GL_LINE_STRIP)
 glVertex2i(x1,y1) ;
 glVertex2i(x2,y2) ;
 glVertex2i(x3,y3) ;
 glVertex2i(x4,y4) ;
glEnd()
```

Line loop

```
glBegin(GL_LINE_LOOP)
 glVertex2i(x1,y1) ;
 glVertex2i(x2,y2) ;
 glVertex2i(x3,y3) ;
 glVertex2i(x4,y4) ;
glEnd()
```

Line details

glLineWidth(GLfloat w) ;

glColor3f(GLfloat r,GLfloat g,GLfloat b) ;

glLineStipple(Glint factor, GLushort pattern) ;

glEnable(GL_LINE_STIPPLE) ;

Polygons in OpenGL

```
glPolygonMode(GL_FRONT,GL_FILL) ;
glPolygonMode(GL_BACK,GL_LINE) ;
glColor3f(red,green,blue) ;
glBegin(GL_POLYGON)
    glNormal3f(v1,v2,v3) ;
    glVertex3f(x1,y1,z1) ;
    …
    glNormal3f(v1n,v2n,v3n) ;
    glVertex3f(xn,yn,zn) ;
glEnd()  ;
```

Higher Primitives in GLUT

```
glutSolidShere() ;
glutSolidCube();
glutSolidCone() ;
glutSolidTeapot() ;
```