

## 15-462 Computer Graphics I

### Lecture 3

## Interaction

Client/Server Model

Callbacks

Double Buffering

Hidden Surface Removal

Simple Transformations

[Angel Ch. 3]

January 22, 2002

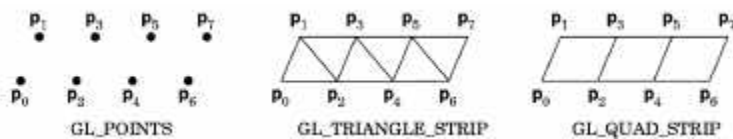
Frank Pfenning

Carnegie Mellon University

<http://www.cs.cmu.edu/~fp/courses/graphics/>

## Surface Orientation (Answer)

- Right-hand rule
- Triangle strip drawn 0-1-2, 2-1-3, 2-3-4, etc.



- All triangles face same direction (here: back)
- Similarly for quad strips 0-1-3-2, 2-3-5-4, etc.

## Choice of Programming Language

- OpenGL lives close to the hardware
- OpenGL is not object-oriented
- OpenGL is not functional
- Use C to expose and exploit low-level details
- Use C++, Java, O'Caml, ... for toolkits
- Support for C and C++ in assignments
- O'Caml anyone? (OpenGL bindings exist!)

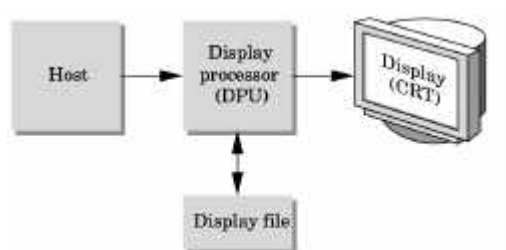
01/22/2002

15-462 Computer Graphics I

3

## Client/Server Model

- Graphics hardware and caching



- Important for efficiency
- Need to be aware where data are stored
- Examples: vertex arrays, display lists

01/22/2002

15-462 Computer Graphics I

4

## Display Lists

- Encapsulate a sequence of drawing commands
- Optimize and store on server

```
GLuint listName = glGenLists(1); /* new name */
glNewList (listName, GL_COMPILE); /* new list */
    glColor3f(1.0, 0.0, 1.0);
    glBegin(GL_TRIANGLES);
        glVertex3f(0.0, 0.0, 0.0);
    ...
    glEnd();
    glTranslatef(1.5, 0.0, 0.0); /* offset next object */
glEndList();
glCallList(listName); /* draw one */
```

01/22/2002

15-462 Computer Graphics I

5

## Display Lists Details

- Useful for sequences of transformations
- Important for complex surfaces
- Another example: fonts
- Hierarchical display lists supported
- Display lists cannot be changed
- Display lists can be replaced
- Not necessary in first assignment

01/22/2002

15-462 Computer Graphics I

6

## Vertex Arrays (Answer)

- Draw cube with  $6*4=24$  or with 8 vertices?
- Expense in drawing and transformation
- Strips help to some extent
- Vertex arrays provide general solution
- Advanced (new in OpenGL 1.2)
  - Define (transmit) array of vertices, colors, normals
  - Draw using index into array(s)
  - Vertex sharing for efficient operations
- Not needed for first assignment

01/22/2002

15-462 Computer Graphics I

7

## Outline

- Client/Server Model
- Callbacks
- Double Buffering
- Hidden Surface Removal
- Simple Transformations
- Example

01/22/2002

15-462 Computer Graphics I

8

## Main Event Loop

- Standard technique for interaction
- Main loop processes events
- Dispatch to functions specified by client
- Callbacks also common in operating systems
- Poor man's functional programming
- Mediates between client and window system

01/22/2002

15-462 Computer Graphics I

9

## Types of Callbacks

- Display ( ): when window must be drawn
- Idle ( ): when no other events to be handled
- Keyboard (unsigned char key, int x, int y): key
- Menu (...): after selection from menu
- Mouse (int button, int state, int x, int y): mouse  
[note error on p.49 of primer]
- Motion (...): mouse movement (see primer)
- Reshape (int w, int h): window resize
- Any callback can be NULL

01/22/2002

15-462 Computer Graphics I

10

## Outline

- Client/Server Model
- Callbacks
- Double Buffering
- Hidden Surface Removal
- Simple Transformations
- Example

01/22/2002

15-462 Computer Graphics I

11

## Screen Refresh

- Common: 60-100 Hz
- Flicker if drawing overlaps screen refresh
- Problem during animation
- Example (cube\_single.c)
- Solution two frame buffers:
  - Draw into one buffer
  - Swap and display, while drawing into other buffer
- Desirable frame rate  $\geq 30$  fps (frames/second)

01/22/2002

15-462 Computer Graphics I

12

## Enabling Modes

- One example of many
- `glutInitDisplayMode (GLUT_SINGLE);`
- `glutInitDisplayMode (GLUT_DOUBLE);`
- `glutSwapBuffers ();`
- If something has no effect, check mode
- Example (cube.c)

01/22/2002

15-462 Computer Graphics I

13

## Outline

- Client/Server Model
- Callbacks
- Double Buffering
- Hidden Surface Removal
- Simple Transformations
- Example

01/22/2002

15-462 Computer Graphics I

14

## Hidden Surface Removal

- Classic problem of computer graphics
- What is visible after clipping and projection?
- Object-space vs image-space approaches
- Object space: depth sort (Painter's algorithm)
- Image space: ray cast (z-buffer algorithm)
- Related: back-face culling

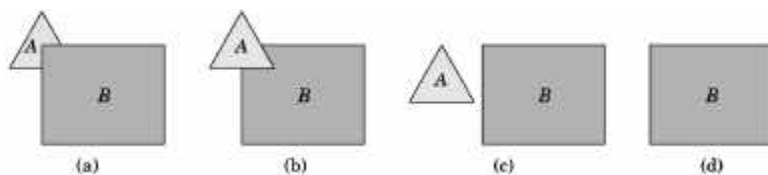
01/22/2002

15-462 Computer Graphics I

15

## Object-Space Approach

- Consider objects pairwise



- Complexity  $O(k^2)$  where  $k = \#$  of objects
- Painter's algorithm: render back-to-front
- "Paint" over invisible polygons
- How to sort and how to test overlap?

01/22/2002

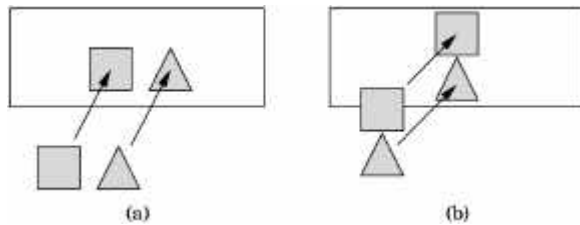
15-462 Computer Graphics I

16



## Depth Sorting

- First, sort by furthest distance  $z$  from viewer
- If minimum depth of A is greater than maximum depth of B, A can be drawn before B
- If either  $x$  or  $y$  extents do not overlap, A and B can be drawn independently



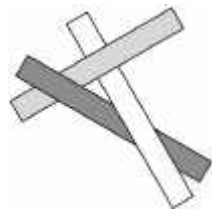
01/22/2002

15-462 Computer Graphics I

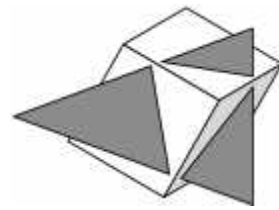
17

## Some Difficult Cases

- Sometimes cannot sort polygons!



Cyclic overlap



Piercing Polygons

- One solution: compute intersections
- Do while rasterizing (difficult in object space)

01/22/2002

15-462 Computer Graphics I

18

## Painter's Algorithm Assessment

- Strengths
  - Simple (most of the time)
  - Handles transparency well
  - Sometimes, no need to sort (e.g., heightfield)
- Weaknesses
  - Clumsy when geometry is complex
  - Sorting can be expensive
- Usage
  - OpenGL (by default)
  - PostScript interpreters

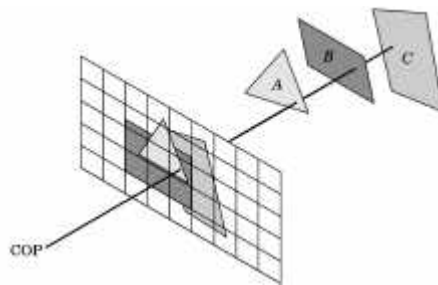
01/22/2002

15-462 Computer Graphics I

19

## Image-Space Approach

- Raycasting: intersect ray with polygons



- $O(k)$  worst case (often better)
- Images can be more jagged

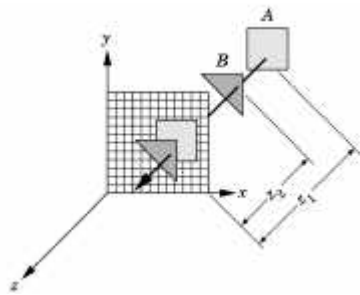
01/22/2002

15-462 Computer Graphics I

20

## The z-Buffer Algorithm

- z-buffer with depth value  $z$  for each pixel
- Before writing a pixel into framebuffer
  - Compute distance  $z$  of pixel origin from viewer
  - If closer write and update z-buffer, otherwise discard



01/22/2002

15-462 Computer Graphics I

21

## z-Buffer Algorithm Assessment

- Strengths
  - Simple (no sorting or splitting)
  - Independent of geometric primitives
- Weaknesses
  - Memory intensive (but memory is cheap now)
  - Tricky to handle transparency and blending
  - Depth-ordering artifacts
- Usage
  - OpenGL when enabled

01/22/2002

15-462 Computer Graphics I

22

## Depth Buffer in OpenGL

- `glutInitDisplayMode(GLUT_DEPTH);`
- `glEnable (GL_DEPTH_TEST);`
- `glClear (GL_DEPTH_BUFFER_BIT);`
- Remember all of these!
- Some “tricks” use z-buffer read-only

01/22/2002

15-462 Computer Graphics I

23

## Outline

- Client/Server Model
- Callbacks
- Double Buffering
- Hidden Surface Removal
- Simple Transformations
- Example

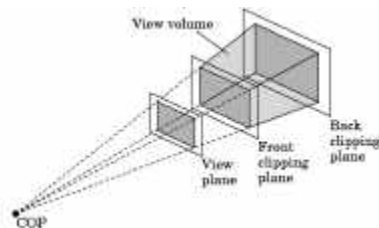
01/22/2002

15-462 Computer Graphics I

24

## Specifying the Viewing Volume

- Clip everything not in viewing volume
  - Separate matrices for transformation and projection
- ```
glMatrixMode (GL_PROJECTION)
glLoadIdentity();
... Set viewing volume ...
glMatrixMode(GL_MODELVIEW)
```



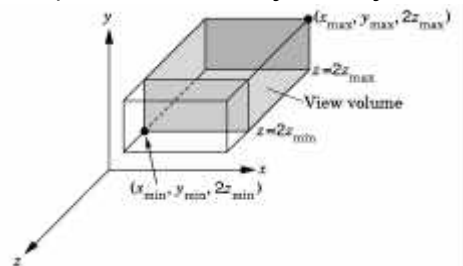
01/22/2002

15-462 Computer Graphics I

25

## Parallel Viewing

- Orthographic projection
- Camera points in negative z direction
- `glOrtho(xmin, xmax, ymin, ymax, near, far)`



- $2z_{\min} = -\text{near}$ ,  $2z_{\max} = -\text{far}$  [diagram correction]

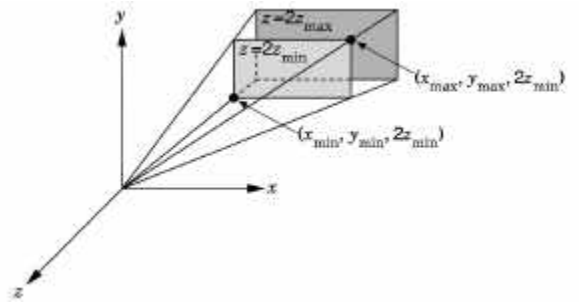
01/22/2002

15-462 Computer Graphics I

26

## Perspective Viewing

- Slightly more complex
- `glFrustum(xmin, xmax, ymin, ymax, near, far)`



- $2z_{\min} = -\text{near}$ ,  $2z_{\max} = -\text{far}$  [diagram correction]

01/22/2002

15-462 Computer Graphics I

27

## Simple Transformations

- Rotate by given angle (in degrees) about ray from origin through  $(x, y, z)$

`glRotate{fd}(angle, x, y, z);`

- Translate by the given x, y, and z values

`glTranslate{fd}(x, y, z);`

- Scale with a factor in the x, y, and z direction

`glScale{fd}(x, y, z);`

01/22/2002

15-462 Computer Graphics I

28

## Outline

- Client/Server Model
- Callbacks
- Double Buffering
- Hidden Surface Removal
- Simple Transformations
- Example

01/22/2002

15-462 Computer Graphics I

29

## Example: Rotating Color Cube

- Adapted from [Angel, Ch. 4]
- Problem:
  - Draw a color cube
  - Rotate it about x, y, or z axis, depending on left, middle or right mouse click
  - Stop when space bar is pressed
  - Quit when q or Q is pressed

01/22/2002

15-462 Computer Graphics I

30

## Step 1: Defining the Vertices

- Use parallel arrays for vertices and colors

```
/* vertices of cube about the origin */
GLfloat vertices[8][3] =
    {{-1.0, -1.0, -1.0}, {1.0, -1.0, -1.0},
     {1.0, 1.0, -1.0}, {-1.0, 1.0, -1.0}, {-1.0, -1.0, 1.0},
     {1.0, -1.0, 1.0}, {1.0, 1.0, 1.0}, {-1.0, 1.0, 1.0}};

/* colors to be assigned to edges */
GLfloat colors[8][3] =
    {{0.0, 0.0, 0.0}, {1.0, 0.0, 0.0},
     {1.0, 1.0, 0.0}, {0.0, 1.0, 0.0}, {0.0, 0.0, 1.0},
     {1.0, 0.0, 1.0}, {1.0, 1.0, 1.0}, {0.0, 1.0, 1.0}};
```

01/22/2002

15-462 Computer Graphics I

31

## Step 2: Set Up

- Enable depth testing and double buffering

```
int main(int argc, char **argv)
{
    glutInit(&argc, argv);
    /* double buffering for smooth animation */
    glutInitDisplayMode
        (GLUT_DOUBLE | GLUT_DEPTH | GLUT_RGB);
    ... /* window creation and callbacks here */
    glEnable(GL_DEPTH_TEST);
    glutMainLoop();
    return(0);
}
```

01/22/2002

15-462 Computer Graphics I

32



## Step 3: Install Callbacks

- Create window and set callbacks

```
glutInitWindowSize(500, 500);  
glutCreateWindow("cube");  
glutReshapeFunc(myReshape);  
glutDisplayFunc(display);  
glutIdleFunc(spinCube);  
glutMouseFunc(mouse);  
glutKeyboardFunc(keyboard);
```

01/22/2002

15-462 Computer Graphics I

33

## Step 4: Reshape Callback

- Enclose cube, preserve aspect ratio

```
void myReshape(int w, int h)  
{  
    GLfloat aspect = (GLfloat) w / (GLfloat) h;  
    glViewport(0, 0, w, h);  
    glMatrixMode(GL_PROJECTION);  
    glLoadIdentity();  
    if (w <= h) /* aspect <= 1 */  
        glOrtho(-2.0, 2.0, -2.0/aspect, 2.0/aspect, -10.0, 10.0);  
    else /* aspect > 1 */  
        glOrtho(-2.0*aspect, 2.0*aspect, -2.0, 2.0, -10.0, 10.0);  
    glMatrixMode(GL_MODELVIEW);  
}
```

01/22/2002

15-462 Computer Graphics I

34

## Step 5: Display Callback

- Clear, rotate, draw, flush, swap

```
GLfloat theta[3] = {0.0, 0.0, 0.0};  
void display(void)  
{ glClear(GL_COLOR_BUFFER_BIT  
      | GL_DEPTH_BUFFER_BIT);  
  glLoadIdentity();  
  glRotatef(theta[0], 1.0, 0.0, 0.0);  
  glRotatef(theta[1], 0.0, 1.0, 0.0);  
  glRotatef(theta[2], 0.0, 0.0, 1.0);  
  colorcube(); glFlush();  
  glutSwapBuffers(); }
```

01/22/2002

15-462 Computer Graphics I

35

## Step 6: Drawing Faces

- Call face(a, b, c, d) with vertex index
- Orient consistently

```
void colorcube(void)  
{  
  face(0,3,2,1);  
  face(2,3,7,6);  
  face(0,4,7,3);  
  face(1,2,6,5);  
  face(4,5,6,7);  
  face(0,1,5,4);  
}
```

01/22/2002

15-462 Computer Graphics I

36

## Step 7: Drawing a Face

- Use vector form of primitives and attributes

```
void face(int a, int b, int c, int d)
{glBegin(GL_POLYGON);
  glColor3fv(colors[a]);
  glVertex3fv(vertices[a]);
  glColor3fv(colors[b]);
  glVertex3fv(vertices[b]);
  glColor3fv(colors[c]);
  glVertex3fv(vertices[c]);
  glColor3fv(colors[d]);
  glVertex3fv(vertices[d]);
glEnd(); }
```

01/22/2002

15-462 Computer Graphics I

37

## Step 8: Animation

- Set idle callback

```
GLfloat delta = 2.0;
GLint axis = 2;
void spinCube()
{
  /* spin cube delta degrees about selected axis */
  theta[axis] += delta;
  if (theta[axis] > 360.0) theta[axis] -= 360.0;

  /* display result */
  glutPostRedisplay();
}
```

01/22/2002

15-462 Computer Graphics I

38

## Step 9: Change Axis of Rotation

- Mouse callback

```
void mouse(int btn, int state, int x, int y)
{
    if (btn==GLUT_LEFT_BUTTON
        && state == GLUT_DOWN) axis = 0;
    if (btn==GLUT_MIDDLE_BUTTON
        && state == GLUT_DOWN) axis = 1;
    if (btn==GLUT_RIGHT_BUTTON
        && state == GLUT_DOWN) axis = 2;
}
```

01/22/2002

15-462 Computer Graphics I

39

## Step 10: Toggle Rotation or Exit

- Keyboard callback

```
void keyboard(unsigned char key, int x, int y)
{
    if (key=='q' || key == 'Q') exit(0);
    if (key==' ') {stop = !stop;};
    if (stop)
        glutIdleFunc(NULL);
    else
        glutIdleFunc(spinCube);
}
```

01/22/2002

15-462 Computer Graphics I

40

## Summary

- Client/Server Model
- Callbacks
- Double Buffering
- Hidden Surface Removal
- Simple Transformations
- Example

01/22/2002

15-462 Computer Graphics I

41

## Announcements

- Please verify access to graphics lab and login
- Check web page for C++ instructions
- Possible guest lecture next Tuesday Jan 29 on Graphics Hardware (Nvidia)
- Nvidia campus visit
  - Monday, Jan 28, 7:00-9:00pm, McKenna Wright
  - Tuesday, Jan 29, Technical Internship Expo

01/22/2002

15-462 Computer Graphics I

42