More on the graphics pipeline
Event driven programming
Nintendo Wii (Michael De Rosa)

Outline
• Double Buffering
• Hidden Surface Removal
• Back-Face Culling

Double Buffering: Screen Refresh
• Solution two frame buffers:
  – Draw into one buffer
  – Swap and display, while drawing into other buffer
• Desirable frame rate >= 30 fps (frames/second)

Review
geometric objects
properties: color...
moves camera and objects around

Double Buffering: Screen Refresh
• Common: 60-100 Hz
• Flicker if drawing overlaps screen refresh
• Problem during animation
• Example (cube_single.c)

Enabling Modes
• glutInitDisplayMode (GLUT_SINGLE);
• glutInitDisplayMode (GLUT_DOUBLE);
• draw your scene
• glutSwapBuffers ();
Outline
- Double Buffering
- Hidden Surface Removal
- Back-Face Culling

Hidden Surface Removal
- What is visible after clipping and projection?

Hidden Surface Removal
- Object-space vs image-space approaches
- Object space: depth sort (Painter’s algorithm)
- Image space: ray cast (z-buffer algorithm)

We’ll get back to this later in the semester in much more detail!

Painters Algorithm
- Render back-to-front
- “Paint” over invisible polygons
- How to sort and how to test overlap?

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

Some Difficult Cases
- Sometimes cannot sort polygons!
- One solution: compute intersections and subdivide
- Do while rasterizing (difficult in object space)
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

**Image-Space Approach**

- Raycasting: intersect ray with polygons
  - O(k) worst case (often better) where k=\# of objects

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

**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 Buffer in OpenGL**

- `glutInitDisplayMode (GLUT_DEPTH);`
- `glEnable (GL_DEPTH_TEST);`
- `glClear (GL_DEPTH_BUFFER_BIT);`

**Outline**

- Double Buffering
- Hidden Surface Removal
- Back-Face Culling
Back-Face Culling

Eliminate polygons not facing the viewer

```c
glCullFace(GL_BACK);
```

More on the graphics pipeline
Event driven programming
Nintendo Wii (Michael De Rosa)

Interactive Graphics Applications

- Games
- Simulation-based training
  - pilot training
- Windows-based programs

Event-driven programming

GUI program should remain idle by default
Do not waste machine resources
Become active only in presence of events
Automatically take care of standard actions

```c
main() {  
    Initialization  
    loop forever  
    wait for next event  
    process event  
}
```

Event-driven programming

GUI program should remain idle by default
Do not waste machine resources
Become active only in presence of events
Automatically take care of standard actions

```c
main() {  
    Initialization  
    loop forever  
    wait for next event  
    process event  
}
```

Control-driven programming

```
main()  
while user does not press quit  
draw cube  
check if mouse pressed  
if mouse pressed  
    change axis of rotation  
check if key pressed  
....
```

Many actions performed by user are non-application specific

Continuous polling of user commands → wasting machine resources
GLUT

- Initialization
  - Open window
  - Set display mode
  - Register callbacks

- Enter main loop
  - glutMainLoop();

Types of Callbacks

- Display (): when window must be drawn
- Idle (): when no other events to be handled
- Keyboard (…): key
- Menu (…): after selection from menu
- Mouse (…): mouse
- Motion (…): mouse movement
- Reshape (int w, int h): window resize
- Any callback can be NULL

Example: Rotating Color Cube

- 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

Step 1: Defining the Vertices

  "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}};

Step 2: Set Up

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

Step 3: Install Callbacks

  Create window and set callbacks
  glutInitWindowSize(500, 500);
  glutCreateWindow("cube");
  glutReshapeFunc(myReshape);
  glutDisplayFunc(display);
  glutIdleFunc(spinCube);
  glutMouseFunc(mouse);
  glutKeyboardFunc(keyboard);
Step 4: Reshape Callback

- Enclose cube, preserve aspect ratio

```c
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);
}
```

Step 5: Display Callback

- Clear, rotate, draw, flush, swap

```c
GLfloat theta[3] = (0.0, 0.0, 0.0);
void display(void)
{
    glClearColor(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();
    glutSwapBuffers();
}
```

Step 6: Drawing Faces

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

```c
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);
}
```

Step 7: Drawing a Face

- Use vector form of primitives and attributes

```c
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();
}
```

Step 8: Animation

- Set idle callback

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

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

Step 9: Change Axis of Rotation

- Mouse callback

```c
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;
}
```
Step 10: Toggle Rotation or Exit

• Keyboard callback

```c
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);
}
```

More on the graphics pipeline
Event driven programming
Nintendo Wii (Michael De Rosa)

Nintendo Wii

• PowerPC CPU @ 729 MHz
• ATI/GPU @ 243 MHz
• 88 MB RAM
• WiFi
• Bluetooth

Wii Controller

• Wiimote
  - Position sensing (via IR triangulation)
  - 3 axis accelerometer
  - 12 buttons
  - Speaker
  - Vibration
  - Nonvolatile storage
• Nunchuk
  - 3 axis accelerometer
  - 2 buttons
  - Analog joystick

Wii Sports Bowling

• Depress B button (starts approach)
• Swing controller back, then forward
• Release B button (releases ball)
• Spin can be applied by tilting the controller

Input Channels
- Accelerometer readings (x6)
- Button state (x14)
- Position
- Joystick angle & deflection
- Wireless connection status
Callbacks for Bowling

- glutDisplayFunc(display); [to display the scene]
- glutKeyboardFunc(keyboard);  [to start/stop the swing]
- glutMotionFunc(mouse); [to control swing params]
- Any others?

The Importance of Interaction Design

- The Wii has significantly less processing/graphics power than the PS3 (and no high def. output)
- Wii sold 600k units over Christmas (vs. 491k for PS3)
- Wii was on the annual “Hot Toy” list
- 05-650 Interface and Interaction Design
- 53-500 Fundamentals of Entertainment Technology

Tech Demo