#4: OpenGL Implementation & Project 2

CSE167: Computer Graphics
TAs: Alex Kozlowski & Cameron Chrisman
UCSD, Winter 2006
How was Project 1?

- Too Hard?
- Too Easy?
- Not enough time?
- Not enough explanation?

Goal of Project 1: *Get comfortable with Matrices, Vectors, Points, and Order of Operations*
Outline for Today

1. What is OpenGL?
2. Program Layout and Common Commands
3. Stack Operations
4. C++, Vectors, Matrices
5. Project 2
What is OpenGL?

- A low-level graphics library specification
- Provides a small set of geometric primitives:
  - points, lines, polygons, images, and bitmaps.
- Support for 2 and 3 dimensions
- Includes commands that control how these objects are rendered into the frame buffer
What is GLUT?

- Like OpenGL, it’s a library
- A windowing API for OpenGL
- Easy for the user to handle window events
  - When to redraw the window
  - Mouse Clicks
  - Keyboard Commands
  - Resizing the window
- Cross platform
Outline for Today

1. What is OpenGL?
2. Program Layout and Common Commands
3. Stack Operations
4. C++, Vectors, Matrices
5. Project 2
int main(int argc, char** argv)
{
    // Initialize glut
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);

    // Window position (from top corner), and size (width and height)
    glutInitWindowSize(360, 360);
    glutCreateWindow("CSE167 Matrix Project");

    // Setup the OpenGL features we'd like to use
    initRendering();

    // Set up callback functions for key presses
    glutKeyboardFunc(myKeyboardFunc); // Handles "normal" ascii symbols

    // Set up the callback function for resizing windows
    glutReshapeFunc(resizeWindow);

    // call this whenever window needs redrawing
    glutDisplayFunc(drawScene);

    // Start the main loop. glutMainLoop never returns.
    glutMainLoop();

    return(0); // This line is never reached.
}
void drawScene(void)
{
    // This command clears the screen to the
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

    // Setup our projection matrix
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluPerspective(60, g_Aspect, .1, 80);

    // Switch to ModelView
    glMatrixMode(GL_MODELVIEW);
    // Camera transformations would typically go here

    // Draw stuff...

    // Swap the buffers
    glutSwapBuffers();

    // Tell glut to immediately redraw the scene for the next frame
    glutPostRedisplay();
}
Drawing Primitives in OpenGL

- Starts with `glBegin(primitive_constant)` and ends with `glEnd()`
- 10 Primitive Types
  - `GL_POINTS`, `GL_LINES`, `GL_LINE_STRIP`, `GL_LINE_LOOP`, `GL_TRIANGLES`, `GL_TRIANGLE_STRIP`, `GL_TRIANGLE_FAN`, `GL_QUADS`, `GL_QUAD_STRIP`, and `GL_POLYGON`
- We’ll use `GL_TRIANGLES` most often in this class
// 2 Triangles
    glBegin(GL_TRIANGLES);
    glColor3f(1,0,0);
    glVertex3f( 1, 1,0);
    glVertex3f(-1, 1,0);
    glVertex3f(-1,-1,0);
    glVertex3f(-1,-1,0);
    glVertex3f( 1,-1,0);
    glVertex3f( 1, 1,0);
    glEnd();

// 1 Quad
    glBegin(GL_QUADS);
    glColor3f(1,0,0);
    glVertex3f( 1, 1,0);
    glVertex3f(-1, 1,0);
    glVertex3f(-1,-1,0);
    glVertex3f( 1,-1,0);
    glEnd();
Outline for Today

1. What is OpenGL?
2. Program Layout and Common Commands
3. Stack Operations
4. C++, Vectors, Matrices
5. Project 2
Books on a Table

LoadIdentity()
PushMatrix()
  Translate(2,1)
drawTable()
PushMatrix()
  Translate(1,1)
  RotateZ(60)
  drawBook()
PopMatrix()
PushMatrix()
  Translate(-1,0)
  RotateZ(-45)
  drawBook()
PopMatrix()
PopMatrix()
Books on a Table

```plaintext
LoadIdentity()
PushMatrix()
  Translate(2,1)
drawTable()
PushMatrix()
  Translate(1,1)
  RotateZ(60)
drawBook()
PopMatrix()
PushMatrix()
  Translate(-1,0)
  RotateZ(-45)
drawBook()
PopMatrix()
PopMatrix()
```

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1
Books on a Table

LoadIdentity()
PushMatrix()
  Translate(2,1)
drawTable()
PushMatrix()
  Translate(1,1)
  RotateZ(60)
drawBook()
PopMatrix()
PushMatrix()
  Translate(-1,0)
  RotateZ(-45)
drawBook()
PopMatrix()
PopMatrix()
Books on a Table

LoadIdentity()
PushMatrix()
    Translate(2,1)
drawTable()
PushMatrix()
    Translate(1,1)
    RotateZ(60)
    drawBook()
PopMatrix()
PushMatrix()
    Translate(-1,0)
    RotateZ(-45)
    drawBook()
PopMatrix()
PopMatrix()
Books on a Table

```
LoadIdentity()
PushMatrix()
  Translate(2,1)
  drawTable()
PushMatrix()
  Translate(1,1)
  RotateZ(60)
  drawBook()
PopMatrix()
PushMatrix()
  Translate(-1,0)
  RotateZ(-45)
  drawBook()
PopMatrix()
PopMatrix()
```
Books on a Table

LoadIdentity()
PushMatrix()
  Translate(2,1)
drawTable()
PushMatrix()
  Translate(1,1)
  RotateZ(60)
  drawBook()
PopMatrix()
PushMatrix()
  Translate(-1,0)
  RotateZ(-45)
  drawBook()
PopMatrix()
PopMatrix()
Books on a Table

LoadIdentity()
PushMatrix()
Translate(2,1)
drawTable()
PushMatrix()
    Translate(1,1)
    RotateZ(60)
drawBook()
PopMatrix()
PushMatrix()
    Translate(-1,0)
    RotateZ(-45)
drawBook()
PopMatrix()
PopMatrix()
Books on a Table

LoadIdentity()
PushMatrix()
  Translate(2,1)
drawTable()
PushMatrix()
  Translate(1,1)
  RotateZ(60)
drawBook()
PopMatrix()
PushMatrix()
  Translate(-1,0)
  RotateZ(-45)
drawBook()
PopMatrix()
PopMatrix()
Books on a Table

LoadIdentity()
PushMatrix()
  Translate(2,1)
drawTable()
PushMatrix()
  Translate(1,1)
  RotateZ(60)
  drawBook()
PopMatrix()
PushMatrix()
  Translate(-1,0)
  RotateZ(-45)
  drawBook()
PopMatrix()
PopMatrix()
Books on a Table

LoadIdentity()
PushMatrix()
  Translate(2,1)
drawTable()
PushMatrix()
  Translate(1,1)
  RotateZ(60)
drawBook()
PopMatrix()
PushMatrix()
  Translate(-1,0)
  RotateZ(-45)
drawBook()
PopMatrix()
PopMatrix()
**Books on a Table**

```plaintext
LoadIdentity()
PushMatrix()
    Translate(2,1)
drawTable()
PushMatrix()
    Translate(1,1)
    RotateZ(60)
drawBook()
PopMatrix()
PushMatrix()
    Translate(-1,0)
    RotateZ(-45)
drawBook()
PopMatrix()
PopMatrix()
```

![Diagram of books on a table with matrix operations]

- \( \text{I*T}(2,1) \)
- \( \text{I*T}(2,1) \)
- \( \text{I} \)
LoadIdentity()
PushMatrix()
    Translate(2,1)
drawTable()
PushMatrix()
    Translate(1,1)
    RotateZ(60)
drawBook()
PopMatrix()
PushMatrix()
    Translate(-1,0)
    RotateZ(-45)
drawBook()
PopMatrix()
PopMatrix()
Books on a Table

LoadIdentity()
PushMatrix()
Translate(2,1)
drawTable()
PushMatrix()
  Translate(1,1)
  RotateZ(60)
  drawBook()
PopMatrix()
PushMatrix()
  Translate(-1,0)
  RotateZ(-45)
  drawBook()
PopMatrix()
PopMatrix()
Books on a Table

LoadIdentity()
PushMatrix()
Translate(2,1)
drawTable()
PushMatrix()
Translate(1,1)
RotateZ(60)
drawBook()
PopMatrix()
PushMatrix()
Translate(-1,0)
RotateZ(-45)
drawBook()
PopMatrix()
PopMatrix()
Outline for Today

1. What is OpenGL?
2. Program Layout and Common Commands
3. Stack Operations
4. C++, Vectors, Matrices
5. Project 2
C++

What's wrong with the following code?

```c++
int *Differences(const int aSamples[], int arraySize) {
    int tempArray[arraySize];
    for(int i = 0; i < arraySize-1; i++){
        tempArray[i] = aSamples[i]-aSamples[i+1];
    }
    return tempArray;
}
```
What’s wrong with the following code?

Fixed.

```cpp
int *Differences(const int aSamples[], int arraySize) {
    int tempArray = new int[arraySize];

    for(int i = 0; i < arraySize-1; i++){
        tempArray[i] = aSamples[i] - aSamples[i+1];
    }

    return tempArray;
}
```
C++

Code runs fine at first, but after a while the computer seems to slow. Why?

```cpp
int main() {  
    while(true) {  
        int array = new int[100];  
        array[0] = 0;  
        array[1] = 1;  
        for(int j = 2; j < 100; j++) {  
            array[j] = array[j-1] + array[j-2];  
        }  
    }  
}
```
C++

Code runs fine at first, but after a while the computer seems to slow. Why?

Fixed.

```cpp
int main() {
    int array = new int[100];
    while(true) {
        array[0] = 0;
        array[1] = 1;
        for(int j = 2; j < 100; j++) {
            array[j] = array[j-1] + array[j-2];
        }
    }
}
```
LocalToWorld Matrix

Assume a spaceship was modeled in local space pointing down the –z axis. It’s placed in the world using the following matrix:

\[
M = \begin{bmatrix}
    a_x & b_x & c_x & d_x \\
    a_y & b_y & c_y & d_y \\
    a_z & b_z & c_z & d_z \\
    0   & 0   & 0  & 1
\end{bmatrix}
\]

- What world space direction is the spaceship facing?
- What is the world space position of the spaceship?
- The pilot of the ship looks up and sees an enemy ship approaching. What world space direction is he looking?
How do you find the angle $\theta$ between vectors $\mathbf{a}$ and $\mathbf{b}$?
Angle Between Vectors

\[ \mathbf{a} \cdot \mathbf{b} = |\mathbf{a}| |\mathbf{b}| \cos \theta \]

\[ \cos \theta = \left( \frac{\mathbf{a} \cdot \mathbf{b}}{|\mathbf{a}| |\mathbf{b}|} \right) \]

\[ \theta = \cos^{-1} \left( \frac{\mathbf{a} \cdot \mathbf{b}}{|\mathbf{a}| |\mathbf{b}|} \right) \]
For an airplane to get a missile locked on, the target must be within a 10 degree cone in front of the plane. If the plane’s matrix is $\mathbf{M}$ and the target position is $\mathbf{t}$, find an expression that determines if the plane can get a lock on.
Target ‘Lock On’

For an airplane to get a missile locked on, the target must be within a 10 degree cone in front of the plane. If the plane’s matrix is $\mathbf{M}$ and the target position is $\mathbf{t}$, find an expression that determines if the plane can get a lock on.

$\mathbf{a} \cdot \mathbf{t}$

*Slide taken from Steve Rotenberg, CSE167 F2005*
Target ‘Lock On’

- We want to check the angle between the heading vector (-c) and the vector from d to t:

\[ \text{if } \cos^{-1} \left( \frac{(d - t) \cdot c}{|d - t|} \right) < 10^\circ \]

- We can speed that up by comparing the cosine instead ( \( \cos(10^\circ) = .985 \) )

\[ \text{if } \frac{(d - t) \cdot c}{|d - t|} > 0.985 \]

*Slide taken from Steve Rotenberg, CSE167 F2005*
Example: Target ‘Lock On’

- We can even speed that up further by removing the division and the square root in the magnitude computation:

\[ if \left( (d - t) \cdot c \right)^2 > 0.970 \cdot |d - t|^2 \]

- All together, this requires 8 multiplications and 8 adds
Alignment to Target

- An object is at position $p$ with a unit length heading of $h$. We want to rotate it so that the heading is facing some target $t$. Find a unit axis $a$ and an angle $\theta$ to rotate around.

Slide taken from Steve Rotenberg, CSE167 F2005
Alignment to Target

\[ a = \frac{h \times (t-p)}{|h \times (t-p)|} \]

\[ \theta = \cos^{-1}\left( \frac{h \cdot (t-p)}{|(t-p)|} \right) \]
Outline for Today

1. What is OpenGL?
2. Program Layout and Common Commands
3. Stack Operations
4. C+++, Vectors, Matrices
5. Project 2
LocalToWorld Matrix

- Implement a stack
- Redo your solar system to use the stack
- Position the Hubble so that it always faces the earth
- Find out whether the sun’s center is within X degrees of the telescope lens