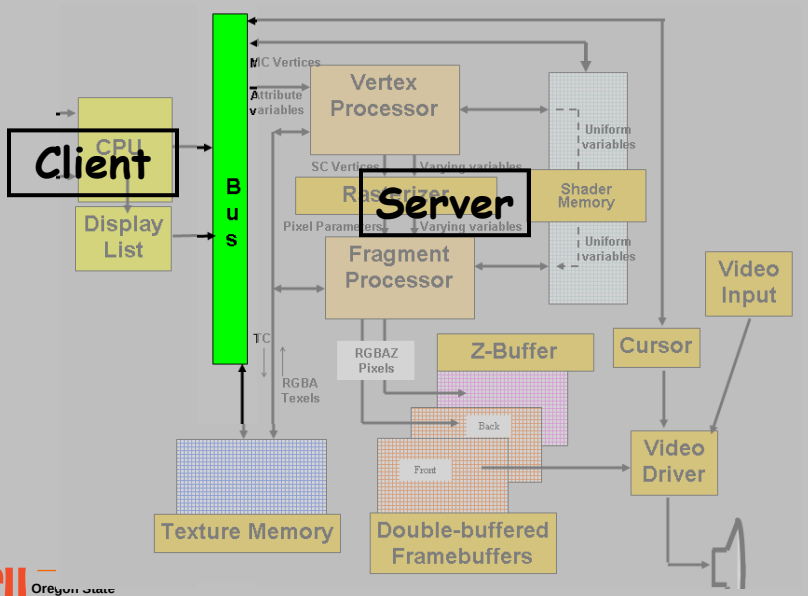
Vertex Arrays

Mike Bailey

Oregon State University

OSU

The OpenGL Client-Server Model



OSU Oregon State University

mjb – January 27, 2007

The Big Idea

- **Store vertex coordinates and vertex attributes in arrays on the host (client).**
- **Transmit the arrays to the graphics card (server), along with indices that tell what vertex numbers need to be connected.**
- **This way, each vertex only needs to be transformed *once*.**
- **It also results in fewer overall function calls.**

Step #1 – Activate the Array Types That You Will Use

glEnableClientState(type)

where type can be:

```
GL_VERTEX_ARRAY
GL_COLOR_ARRAY
GL_NORMAL_ARRAY
GL_SECONDARY_COLOR_ARRAY
GL_TEXTURE_COORD_ARRAY
```

- Call this as many times as you need to enable all the arrays that you will need.
- There are other types, too.
- To deactivate a type, call:

glDisableClientState(type)

Step #2 – Fill the Arrays

```
static GLfloat Vertices[ ][3] = {
{
        { 1., 2., 3. },
        { 4., 5., 6. },
        . . .
};
```


Step #3 – Specify the Arrays

```
glVertexPointer( size, type, stride, array );
glColorPointer( size, type, stride, array );
glNormalPointer( type, stride, array );
glSecondaryColorPointer( size, type, stride, array );
glTexCoordPointer( size, type, stride, array );
```

size can be: 2, 3, or 4

type can be:

```
GL_SHORT
GL_INT
GL_FLOAT
GL_DOUBLE
```

stride is the byte offset between consecutive entries in the array (0 means tightly packed)

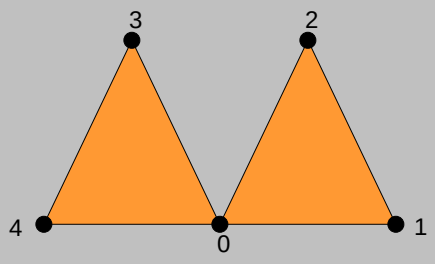
array is the name of the corresponding data array

Step #4 – Specify the Connections

List the vertices individually:

```
glBegin( GL_TRIANGLES );
        glArrayElement( 0 );
        glArrayElement( 1 );
        glArrayElement( 2 );

        glArrayElement( 0 );
        glArrayElement( 3 );
        glArrayElement( 4 );
glEnd( );
```

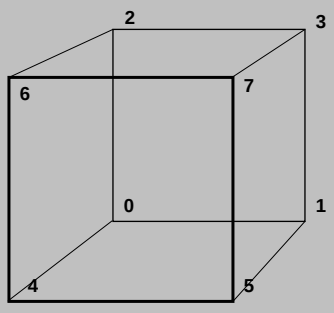


List the vertices together:

```
Static GLuint TriIndices[][3] =
{
        { 0, 1, 2 },
        { 0, 3, 4 }
};

glDrawElements( GL_TRIANGLES, 6, GL_UNSIGNED_INT, TriIndices );
```


Cube Example

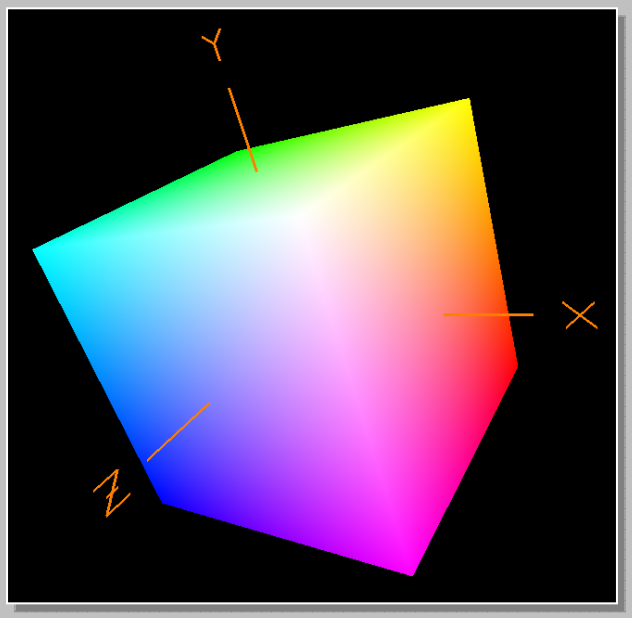



```
static GLfloat CubeVertices[ ][3] =
{
        { -1., -1., -1. },
        {  1., -1., -1. },
        { -1.,  1., -1. },
        {  1.,  1., -1. },
        { -1., -1.,  1. },
        {  1., -1.,  1. },
        { -1.,  1.,  1. },
        {  1.,  1.,  1. }
};
```

```
static GLfloat CubeColors[ ][3] =
{
        { 0., 0., 0. },
        { 1., 0., 0. },
        { 0., 1., 0. },
        { 1., 1., 0. },
        { 0., 0., 1. },
        { 1., 0., 1. },
        { 0., 1., 1. },
        { 1., 1., 1. },
};
```

```
static GLuint CubeIndices[ ][4] =
{
        { 0, 2, 3, 1 },
        { 4, 5, 7, 6 },
        { 1, 3, 7, 5 },
        { 0, 4, 6, 2 },
        { 2, 6, 7, 3 },
        { 0, 1, 5, 4 }
};
```

Cube Example



Cube Example – glArrayElement( ) calls

```
glEnableClientState( GL_VERTEX_ARRAY );
glEnableClientState( GL_COLOR_ARRAY );
glVertexPointer( 3, GL_FLOAT, 0, CubeVertices );
glColorPointer(   3, GL_FLOAT, 0, CubeColors );
glBegin( GL_QUADS );
        glArrayElement( 0 );
        glArrayElement( 2 );
        glArrayElement( 3 );
        glArrayElement( 1 );
        glArrayElement( 4 );
        glArrayElement( 5 );
        glArrayElement( 7 );
        glArrayElement( 6 );
        glArrayElement( 1 );
        glArrayElement( 3 );
        glArrayElement( 7 );
        glArrayElement( 5 );
        glArrayElement( 0 );
        glArrayElement( 4 );
        glArrayElement( 6 );
        glArrayElement( 2 );
        glArrayElement( 2 );
        glArrayElement( 6 );
        glArrayElement( 7 );
        glArrayElement( 3 );
        glArrayElement( 0 );
        glArrayElement( 1 );
        glArrayElement( 5 );
        glArrayElement( 4 );
glEnd( );
```

Cube Example – glDrawElements( ) call

```
glEnableClientState( GL_VERTEX_ARRAY );
glEnableClientState( GL_COLOR_ARRAY );

glVertexPointer( 3, GL_FLOAT, 0, CubeVertices );
glColorPointer(  3, GL_FLOAT, 0, CubeColors );

glDrawElements( GL_QUADS, 24, GL_UNSIGNED_INT, CubeIndices );
```