


Optimizations in OpenGL

- Graphics is compute intensive
 - Lots of trigonometry, etc.
- Speeding up execution is common
- Three techniques
 - Display lists
 - Vertex arrays
 - Hardware accelerators [not covered]


1



Display Lists

- Mechanism for caching pre-computed OpenGL calls
- Faster than immediate mode calls
 - Only one call to execute many commands
 - Geometry is already "computed"
- Higher initial overhead
 - Display list calls are more expensive

2



Display List Basics

- Identified by a positive int
 - `GLuint`
 - 0 is **NOT** a valid display list
- Framed by special calls
 - `glNewList`
 - `glEndList`

3

Example Display List

```

GLuint mylist = glGenLists(1);
if (mylist) {
    glNewList(mylist, GL_COMPILE);
    ...
    // OpenGL calls
    ...
    glEndList();
}
    
```

How many?

GL_COMPILE - Create list only
 GL_COMPILE_AND_EXECUTE - Create list and execute commands

4

Display List Limitations


- Display lists are entirely pre-computed
 - Parameters are evaluated at creation time
 - Even array parameters (the pointer is not stored, the data in the array is)
- Not all OpenGL calls are stored
 - Any call returning a value through a by-reference (pointer) parameter
 - Anything affecting client state

5

Invalid Display List Calls

glColorPointer	glFlush	glNormalPointer
glTexCoordPointer	glGenLists	glPixelStore
glDisableClientState	glGet*	glReadPixels
glEdgeFlagPointer	glRenderMode	glIndexPointer
glEnableClientState	glFinish	glSelectBuffer
glFeedbackBuffer	glIsEnabled	glDeleteLists
glInterleavedArrays	glIsList	glVertexPointer


6



Display List Management

- `glCallList`
 - Execute a display list
- `glIsList`
 - Is this list being used?
- `glDeleteLists`
 - Destroy a consecutive set of lists
- `glNewList`
 - If you reuse a list index the old list is deleted


7



Compound Display Lists

- Display lists may be nested
 - One list can invoke others
 - Maximum depth is at least 64
 - `GL_MAX_LIST_NESTING`
- Display lists can be indexed
 - `glListBase` – Determines called lists relative to this base when `glCallLists` is used

8



Display Lists and OpenGL State

- State changes persist after a executing list
- To preserve state information use the stack
 - `glPushMatrix, glPopMatrix`
 - `glPushAttrib, glPopAttrib`
- Often a complex choice
 - Remember: You cannot pass parameters to a display list

9

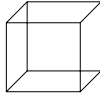
Typical Applications

- Complex, reusable geometry
 - Set state information
 - e.g. Matrices, color, etc.
 - Invoke the geometry via a display list
- Complex state changes
 - Encode expensive state changes into a faster display list

10

Vertex Arrays

- Most 3-D objects have redundancies
- Consider a cube
 - 6 faces
 - 8 vertices
 - 12 edges
 - A naive drawing strategy might specify each vertex up to 3 times!




11

Polygon/Polyhedron Tables (CS321)

- Storing a polyhedron
 - Collection of vertices
 - Only unique for convex polyhedra
 - More realistic: 3 lists
 - Vertex list – 3-D points
 - Edge list – pairs of vertex pointers
 - Surface list – in order edge lists


12



Using Vertex Arrays

1. Enable up to 6 arrays
 - Vertices, color (2), normals, texture, edge flags
2. Put data in the arrays
3. Access the data
 - By index
 - By list
 - In order


13



Enabling the Arrays

- `glEnableClientState`
 - `GL_VERTEX_ARRAY`
 - `GL_COLOR_ARRAY`
 - `GL_INDEX_ARRAY`
 - `GL_NORMAL_ARRAY`
 - `GL_TEXTURE_COORD_ARRAY`
 - `GL_EDGE_FLAG_ARRAY`
- `glDisableClientState`

14



Putting Data in the Arrays

- `gl_____Pointer` – Specify array data
 - `_____` = Array to use (*e.g.*, "Vertex")
 - Number of elements per entry
 - Type of data in each element (*e.g.*, `GL_DOUBLE`)
 - Bytes between consecutive entries (stride); 0 – tightly packed (interleaving, p. 72)
 - Pointer (array) of data

15

Accessing Data in the Arrays (1)

- Simple indexing
 - `glArrayElement`
 - Get indexed element of **all** active arrays
- By sequential list
 - `glDrawArrays`
 - Get consecutive indexed elements

16

Accessing Data in the Arrays (2)

- By indexed list (the most versatile)
 - `glDrawElements`
 - Provide an array of indexes
 - Specify how interpreted
 - i.e. How the vertices are linked
 - Can't be used inside a `glBegin/glEnd`
 - Mode (e.g., `GL_QUADS`) is 1st argument

17
