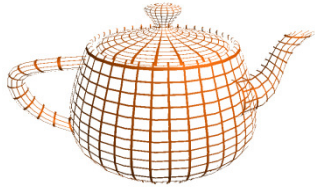




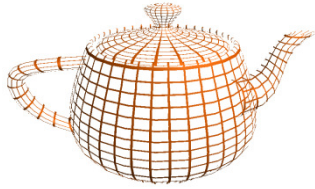
Computação Gráfica

Desempenho



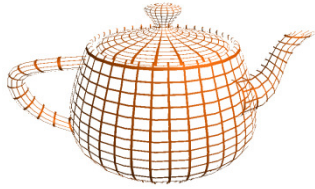
Desempenho

- Tópicos:
 - Display Lists
 - Vertex Buffer Objects



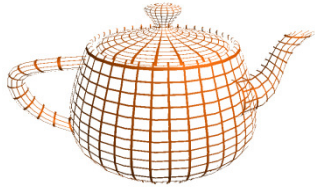
Display Lists

- As DL são um mecanismo que permite "compilar" no driver sequências de comandos. Por exemplo:
 - Transformações geométricas
 - Coordenadas de Vértices
 - Coordenadas de texturas e normais
 - Mudanças de estado, ex: iluminação e cores.



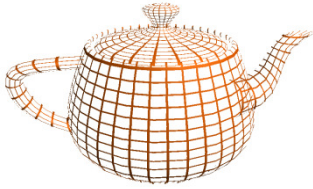
Display Lists

- Exemplos:
 - Ao invocar `glRotate` no modo imediato, é necessário construir a matriz respectiva.
 - Caso `glRotate` seja incluído na DL, então é possível construir a matriz ao criar a DL, na execução só é executada a multiplicação de matrizes.
 - Um display list pode também concatenar matrizes resultantes de transformações geométricas durante a sua criação.



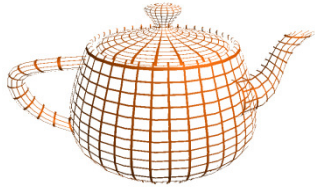
Display Lists

- Vantagens:
 - As DLs são armazenadas na memória da placa, se possível.
 - Uma só invocação versus n para cada um dos comandos armazenados.



Display Lists

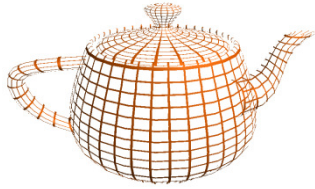
- Uma *display list* é estática.
 - => Não é possível editar os comandos de uma *display list*.
 - Motivo: A edição de uma DL podia provocar fragmentações de memória, e implicava uma gestão de memória mais exigente.



Display Lists

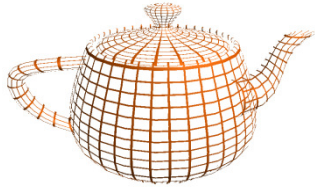
- Display Lists Hierárquicas
 - Uma *display list* pode conter referências a outras *display lists*.
 - A recursividade não é permitida.
 - O nível de aninhamento, segundo a especificação, pode ir até 64.
 - Para verificar o nível máximo de alinhamento permitido:

```
glGetIntegerv(GL_MAX_LIST_NESTING, GLint *maxlevel);
```



Display Lists

- Display Lists Hierarquicas (cont.)
 - Permitem uma forma básica de edição de conteúdos.
 - Ao alterar uma *display list* previamente aninhada, a “mãe” também é implicitamente alterada.

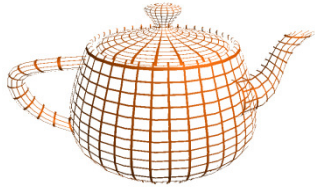


Display Lists

- Setup

`GLuint glGenLists(GLsizei range);`

Gera um conjunto de listas sequenciais. Devolve o índice da primeira lista.



Display Lists

- Definição

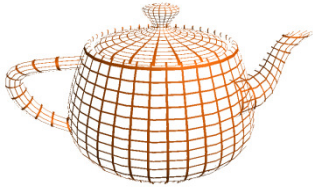
```
void glNewList( GLuint list,  GLenum mode );
```

```
    // comandos para a lista
```

```
void glEndList( void );
```

list: o índice da lista

mode: `GL_COMPILE` ou `GL_COMPILE_AND_EXECUTE`



Display Lists

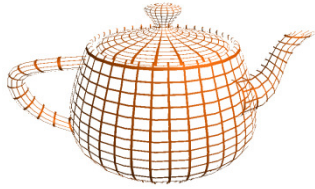
- Utilização

glCallList(GLuint *list*);

realiza o render de uma *display list* com o índice *list*.

glCallLists(Glsizei, n, GLenum type, const Glvoid *lists);

render de um conjunto de *display lists*, cujos índices estão especificados no *array lists*.

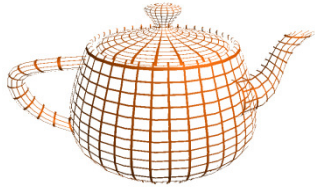


Display Lists

- Libertar recursos

```
void glDeleteLists(GLuint listID, GLsizei numberOfLists);
```

Liberta recursos de uma sequência de comprimento *numberOfLists*, a partir do índice *listID*.



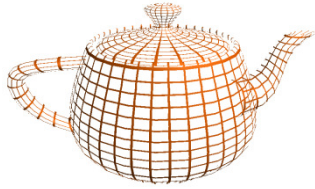
Display Lists

Setup e Geração

```
id = GenLists(1);  
glNewList(id, GL_COMPILE);  
    glTranslatef(10.0f, 5.0f, 3.0f);  
    glBegin(GL_TRIANGLE_STRIP);  
        glVertex3f(1.0f, 1.0f, 1.0f);  
    ...  
    glEnd();  
glEndList();
```

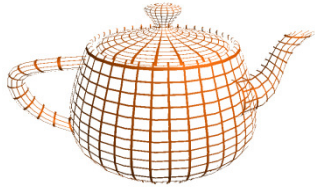
Utilização

```
glCallList(id);
```



Display Lists

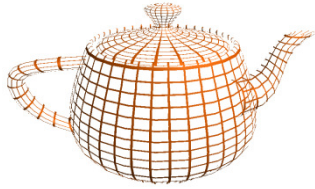
```
void drawSnowMan() {
    glColor3f(1.0f, 1.0f, 1.0f);
    // Draw Body
    glTranslatef(0.0f ,0.75f, 0.0f); glutSolidSphere(0.75f,20,20);
    // Draw Head
    glTranslatef(0.0f, 1.0f, 0.0f); glutSolidSphere(0.25f,20,20);
    // Draw Eyes
    glPushMatrix();
    glColor3f(0.0f,0.0f,0.0f);
    glTranslatef(0.05f, 0.10f, 0.18f); glutSolidSphere(0.05f,10,10);
    glTranslatef(-0.1f, 0.0f, 0.0f); glutSolidSphere(0.05f,10,10);
    glPopMatrix();
    // Draw Nose
    glColor3f(1.0f, 0.5f , 0.5f);
    glRotatef(0.0f,1.0f, 0.0f, 0.0f); glutSolidCone(0.08f,0.5f,10,1);
}
```



OpenGL - Display Lists

- Sem Display Lists

```
for(i = -3; i < 3; i++)  
    for(int j=-3; j < 3; j++) {  
        glPushMatrix();  
        glTranslatef(i*10.0,0,j * 10.0);  
        drawSnowMan();  
        glPopMatrix();  
    }
```



OpenGL - Display Lists

- Display List para o boneco

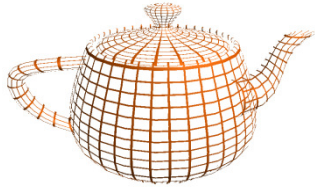
Setup

```
snowManDL = glGenLists(1);
```

```
glNewList(snowManDL, GL_COMPILE);
```

```
    drawSnowMan();
```

```
glEndList();
```

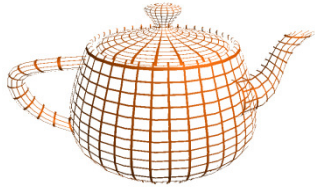



OpenGL - Display Lists

- Display List para o boneco

Utilização

```
for(i = -3; i < 3; i++)  
    for(int j=-3; j < 3; j++) {  
        glPushMatrix();  
        glTranslatef(i*10.0,0,j * 10.0);  
        glCallList(snowManDL);  
        glPopMatrix();  
    }
```



OpenGL - Display Lists

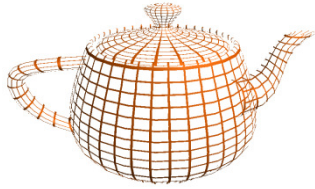
- Display List com todos os bonecos de neve

Setup

```
glNewList(loopDL, GL_COMPILE);
for(int i = -3; i < 3; i++)
    for(int j = -3; j < 3; j++) {
        glPushMatrix();
        glTranslatef(i*10.0, 0, j * 10.0);
        drawSnowMan();
        glPopMatrix();
    }
glEndList();
```

Utilização

```
glCallList(loopDL);
```



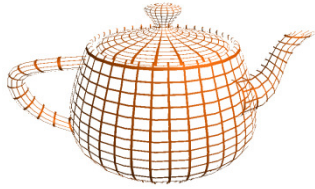
OpenGL - Display Lists

- *Display List Hierárquica*

Setup

```
loopDL = glGenLists(2);
glNewList(loopDL+1, GL_COMPILE);
    drawSnowMan();
glEndList();

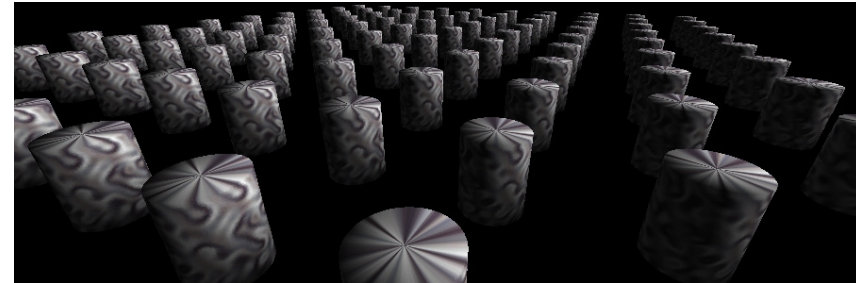
glNewList(loopDL, GL_COMPILE);
for(int i = -3; i < 3; i++)
    for(int j = -3; j < 3; j++) {
        glPushMatrix();
        glTranslatef(i*10.0, 0, j * 10.0);
        glCallList(loopDL+1);
        glPopMatrix();
    }
glEndList();
```



Display Lists

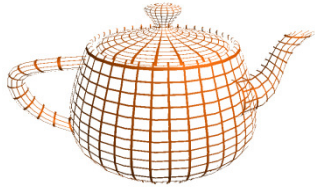
- Teste

- 100 cilindros



- número de triângulos de cada cilindro =
{60,600,6000,60000}

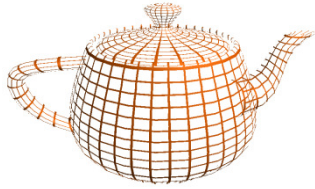
- Pentium 4 2.53Ghz, 1GB DDR 400, GeForce TI
4800 SE



Display Lists

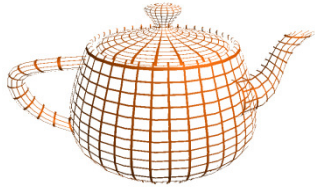
- Resultados

	<i>6000</i>	<i>60.000</i>	<i>600.000</i>	<i>6.000.000</i>
<i>Sem DL</i>	1106	212	22.77	2.17
<i>DL simples</i>	1227	250	77.53	7.91
<i>DL Hierárquica</i>	1227	251	77.53	7.91
<i>DL Total</i>	1581	490	66.01	2.53



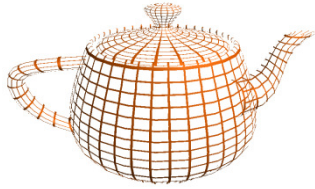
Desempenho

- Tópicos:
 - Display Lists
 - Vertex Buffer Objects



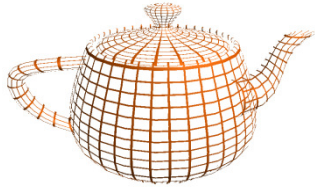
Vertex Buffer Objects

- São necessárias $3n$ invocações de funções para submeter n triângulos com coordenadas de textura e normais.
- Objectivo: eliminar o tempo de submissão
- Processo:
 - Definir arrays para coordenadas, coordenadas de textura e normais
 - Submeter os arrays uma única vez, ficando estes a residir na placa gráfica
 - Desenhar com uma única invocação



Vertex Buffer Objects

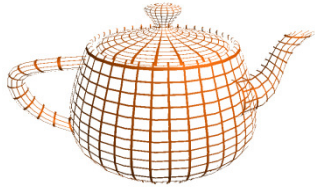
- Permitem armazenar os vértices de forma otimizada, minimizando a memória gráfica necessária.
- Exemplo: desenhar um terreno com *triangle strips*
- $n-1$ strips $\Rightarrow (n-1) * 2n$ vértices submetidos
- em array pode-se enviar somente $n*n$ vértices e construir as strips através de um array de índices.



Vertex Buffer Objects

- Passo 1 - Criação dos arrays e activar funcionalidade
 - alocar arrays para vértices, normais, ...
 - Activar Buffers

```
glEnableClientState(GL_VERTEX_ARRAY);  
glEnableClientState(GL_NORMAL_ARRAY);
```

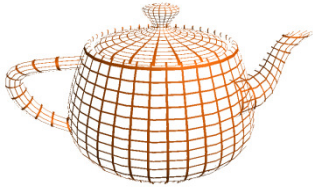


Vertex Buffer Objects

- **Passo 2: Gerar VBOs**

```
GLuint buffers[n];  
float *vertexB, *normalB; // arrays de vértices,  
    normais, etc...  
  
...  
glGenBuffers(n, buffers);  
glBindBuffer(GL_ARRAY_BUFFER, buffers[0]);  
glBufferData(GL_ARRAY_BUFFER, arraySize, vertexB,  
    GL_STATIC_DRAW);  
  
glBindBuffer(GL_ARRAY_BUFFER, buffers[1]);  
glBufferData(GL_ARRAY_BUFFER, arraySize, normalB,  
    GL_STATIC_DRAW);
```

em bytes

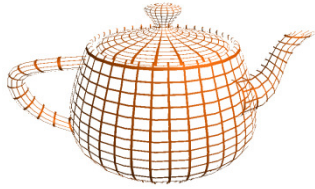


Vertex Buffer Objects

- Passo 3: Atribuir Semântica
 - Indicar para cada buffer qual a sua utilização
 - Define quais os arrays a serem desenhados

```
glBindBuffer(GL_ARRAY_BUFFER, buffers[0]);  
glVertexAttribPointer(3, GL_FLOAT, 0, 0);
```

```
glBindBuffer(GL_ARRAY_BUFFER, buffers[1]);  
glNormalPointer(GL_FLOAT, 0, 0);
```

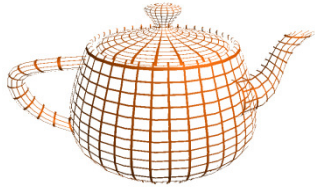


Vertex Buffer Objects

- Passo 4 : Desenhar com VBOs

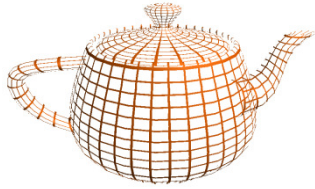
```
glDrawArrays(mode, first, count);
```

```
mode: GL_TRIANGLES, GL_TRIANGLE_STRIP ...
```



Vertex Buffer Objects

- Desta forma enviam-se tantos vértices como no modo imediato.
- Através da utilização de índices é possível reutilizar vértices.
- Os índices também podem ser armazenados na placa.

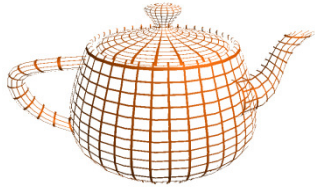


Vertex Buffer Objects

- Define-se um array para os índices dos vértices a desenhar.

```
glDrawElements(modo, count, tipo, índices);
```

- modo: `GL_TRIANGLE_STRIP`, ...



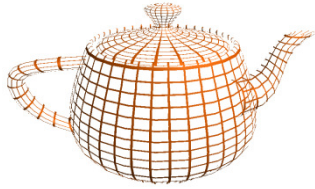
Vertex Buffer Objects

- Os índices também podem residir na placa:

```
glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, buffers[2]);  
glBufferData(GL_ELEMENT_ARRAY_BUFFER, size,  
pointer, GL_STATIC_DRAW);
```

Para desenhar:

```
glDrawElements(modos, count, tipo, 0);
```



Referências

- OpenGL Reference Manual, OpenGL Architecture Review Board.