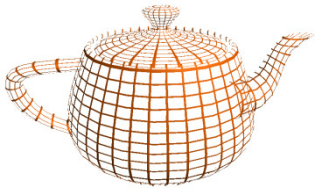
sim

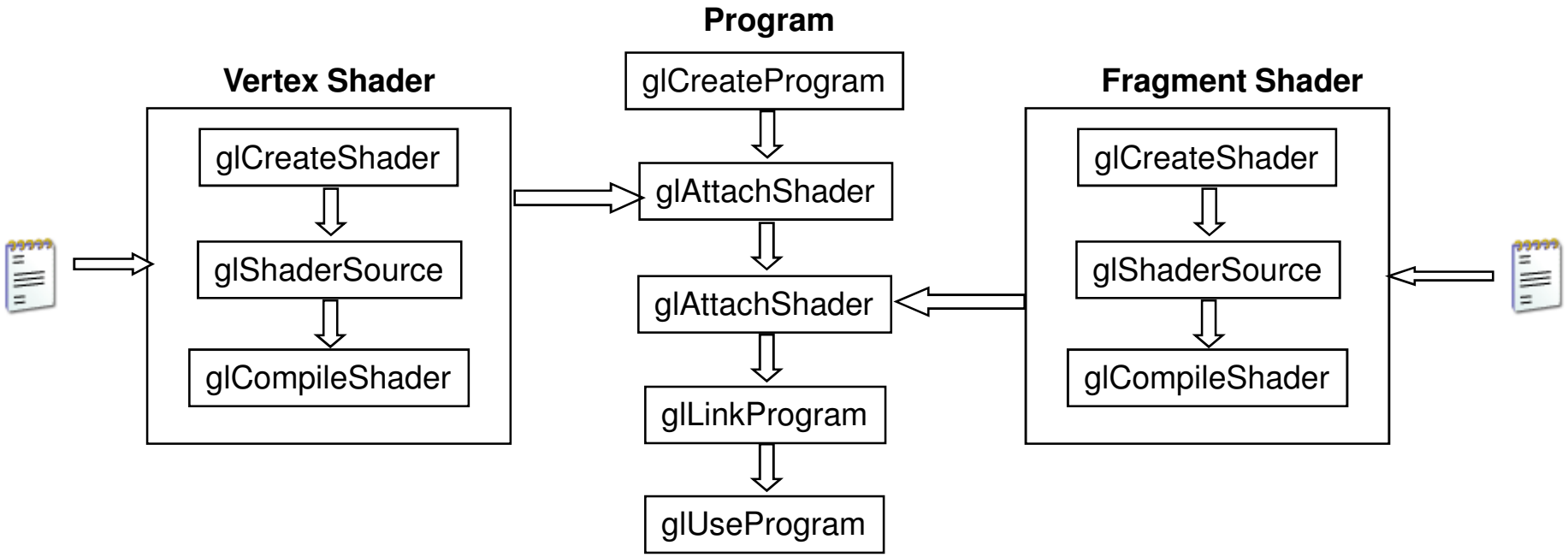
Computação Gráfica

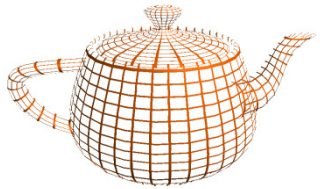
OpenGL e GLSL



GLSL - Aplicação OpenGL

- OpenGL Setup





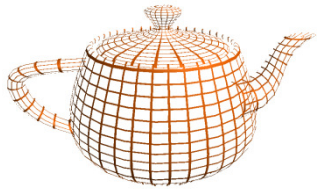
GLSL - Aplicação OpenGL

- Protótipos de funções:

```
GLuint glCreateShader (enum type)
```

  - Devolve um handler para o shader
  - Recebe como parâmetro o tipo de shader:

```
GL_VERTEX_SHADER
GL_FRAGMENT_SHADER
```

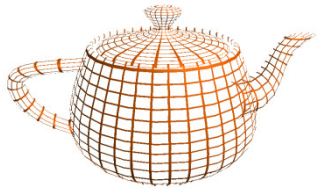


GLSL - Aplicação OpenGL

- Protótipos de funções:

```
GLuint glShaderSource(uint shader, sizei count,
  const char **string, const int *length)
```

  - shader: O shader ao qual se destina o código. Este valor é o devolvido pela função CreateShader.
  - count: número de strings
  - string: array de strings
  - length: dimensão de cada string, ou NULL, no caso das strings serem terminadas com '\0'.

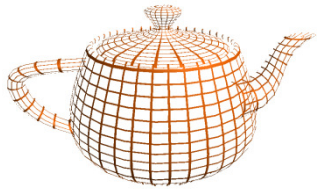


GLSL - Aplicação OpenGL

- Protótipos de funções:

```
void glCompileShader(uint shader)
```

- O resultado da compilação pode ser inquirido através da função GetShaderiv. Os detalhes encontram-se no infoLog associado a cada shader.

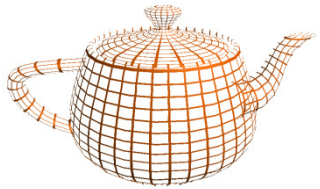


GLSL - Aplicação OpenGL

- Protótipos de funções:

```
uint glCreateProgram()
```

- Devolve um handler para um Program Object.

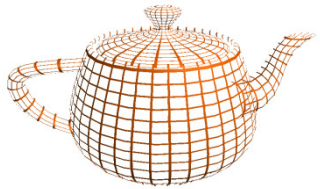


GLSL - Aplicação OpenGL

- Protótipos de funções:

```
void glAttachShader(uint program, uint shader)
```

- Função que liga um shader a um programa.
- Um programa pode ter vários shaders do mesmo tipo, no entanto só um deles pode ter uma função main.
- Um shader pode também estar associado a vários programas.
- Esta operação não necessita de ser realizada com um shader compilado, ou até com código, podendo ser efectuada com um shader recém-criado.

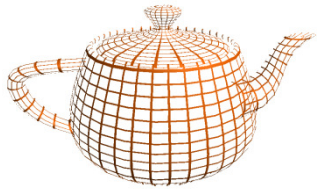


GLSL - Aplicação OpenGL

- Protótipos de funções:

```
void glLinkProgram (uint programa)
```

- Realiza a operação sobre os shaders ligados ao programa. O estado da operação pode ser consultado através da função GetProgramiv. Os detalhes encontram-se no infoLog associado a cada programa.



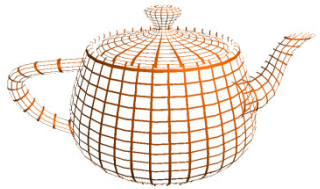
GLSL - Aplicação OpenGL

- Protótipos de funções:

```
void glUseProgram (uint programa)
```

- Se (programa != 0) a funcionalidade das componentes programáveis passa a ser determinado pelo programa.

- Se (programa == 0) o sistema gráfico passa a utilizar a funcionalidade fixa



GLSL - Aplicação OpenGL

- Código para criação de shaders

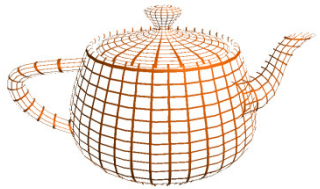
```
GLuint v,f;

v = glCreateShader (GL_VERTEX_SHADER);
f = glCreateShader (GL_FRAGMENT_SHADER);

vs = textFileRead("stripes.vert");
fs = textFileRead("stripes.frag");

glShaderSource (v, 1, &vs,NULL);
glShaderSource (f, 1, &fs,NULL);

glCompileShader (v);
glCompileShader (f);
```



GLSL - Aplicação OpenGL

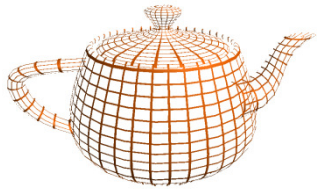
- Construção do programa

```
...
p = glCreateProgram ();

glAttachShader(p,v);
glAttachShader(p,f);

glLinkProgram (p);

glUseProgram(p);
```



GLSL - Aplicação OpenGL

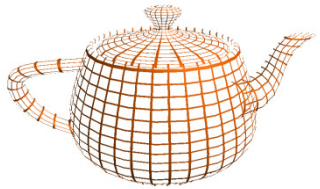
- Análise dos resultados da compilação e link

```
void glGetShaderiv(GLuint object, GLenum type, int *param);
```

- Parameters:
  - object - the handler to the object. Either a shader or a program
  - type - GL_COMPILE_STATUS.
  - param - the return value, GL_TRUE if OK, GL_FALSE otherwise.

```
void glGetProgramiv(GLuint object, GLenum type, int *param);
```

- Parameters:
  - object - the handler to the object. Either a shader or a program
  - type - GL_LINK_STATUS.
  - param - the return value, GL_TRUE if OK, GL_FALSE otherwise.



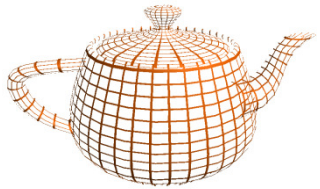
GLSL - Aplicação OpenGL

- InfoLog
  - A seguinte função dá acesso ao texto do log:

```
void glGetShaderInfoLog(GLuint object, int maxLen, int *len,
   char *log);
void glGetProgramInfoLog(GLuint object, int maxLen, int *len,
   char *log);
```

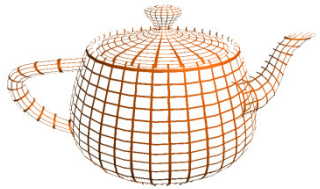
  - Para determinar o parâmetro **infoLogLength** invoca-se a seguinte função:

```
void glGetShaderiv(GLuint object, GLenum type, int *param);
void glGetProgramiv(GLuint object, GLenum type, int *param);
```



GLSL - Aplicação OpenGL

- Parâmetros Uniform
  - São parâmetros que não variam ao longo da primitiva, ou seja, não devem ser modificados entre glBegin e glEnd.
  - Ex: Posição da luz, índice de refracção, etc...



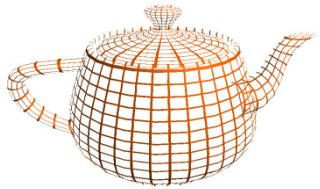
GLSL - Aplicação OpenGL

- Parâmetros Uniform
  - Setup: Determinar a sua "localização", ou seja um índice que nos permitirá posteriormente aceder ao parâmetro

```
int loc = glGetUniformLocation (GLuint program, char *name);
```

  - Utilização: Afectar o valor do parâmetro faz-se, por exemplo, através da seguinte função:

```
void  glUniform1f (loc, float a);
```



Exercício prático

- Continuar o desenvolvimento da aplicação do cenário de forma a incorporar shaders:
- Implementar a utilização do shader fornecido .
- Passar parâmetros ao Shader.
- Alterar o shader de forma a utilizar como cor base a cor que é fornecida pela aplicação OpenGL.