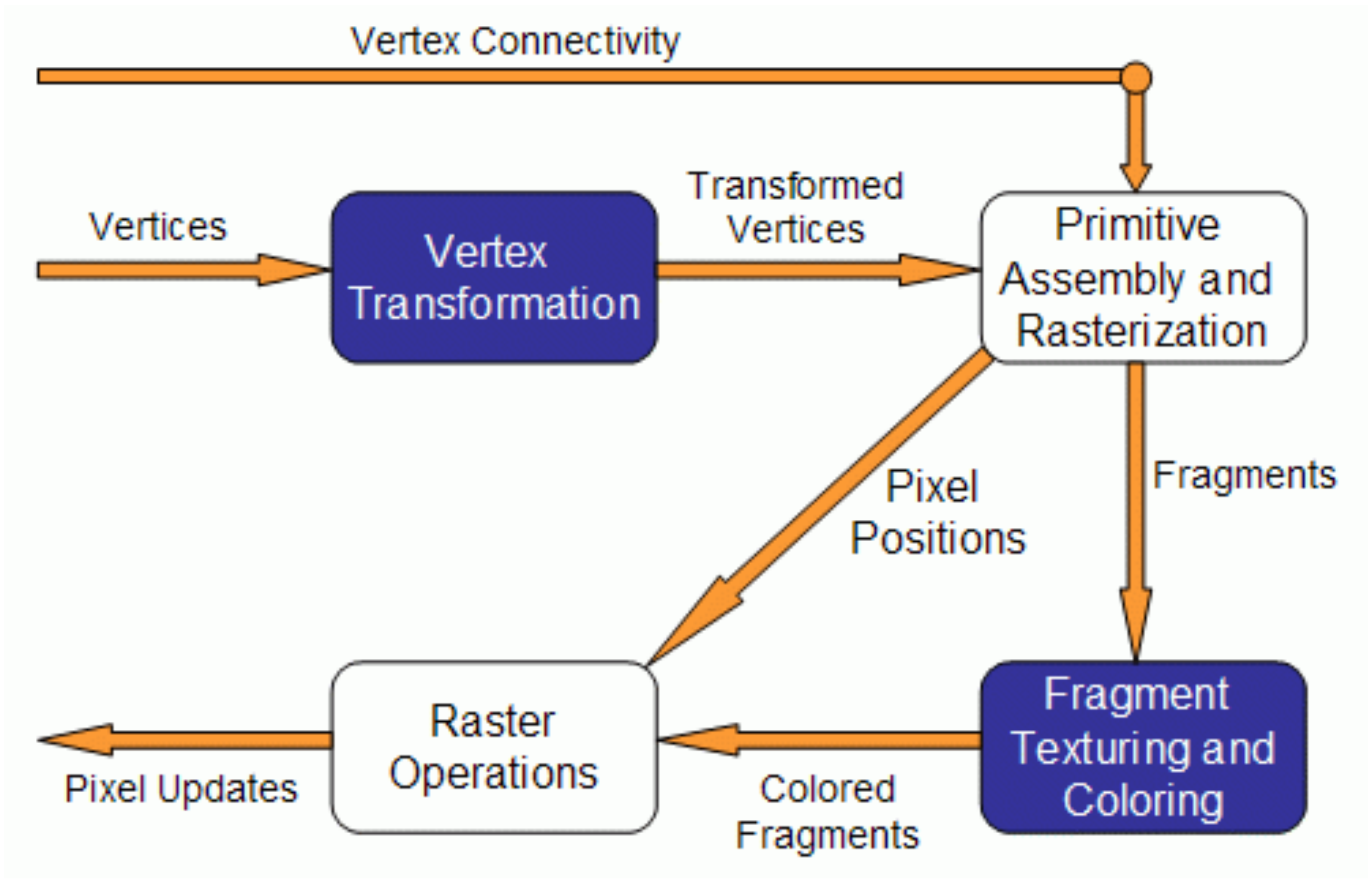
COMPUTER GRAPHICS

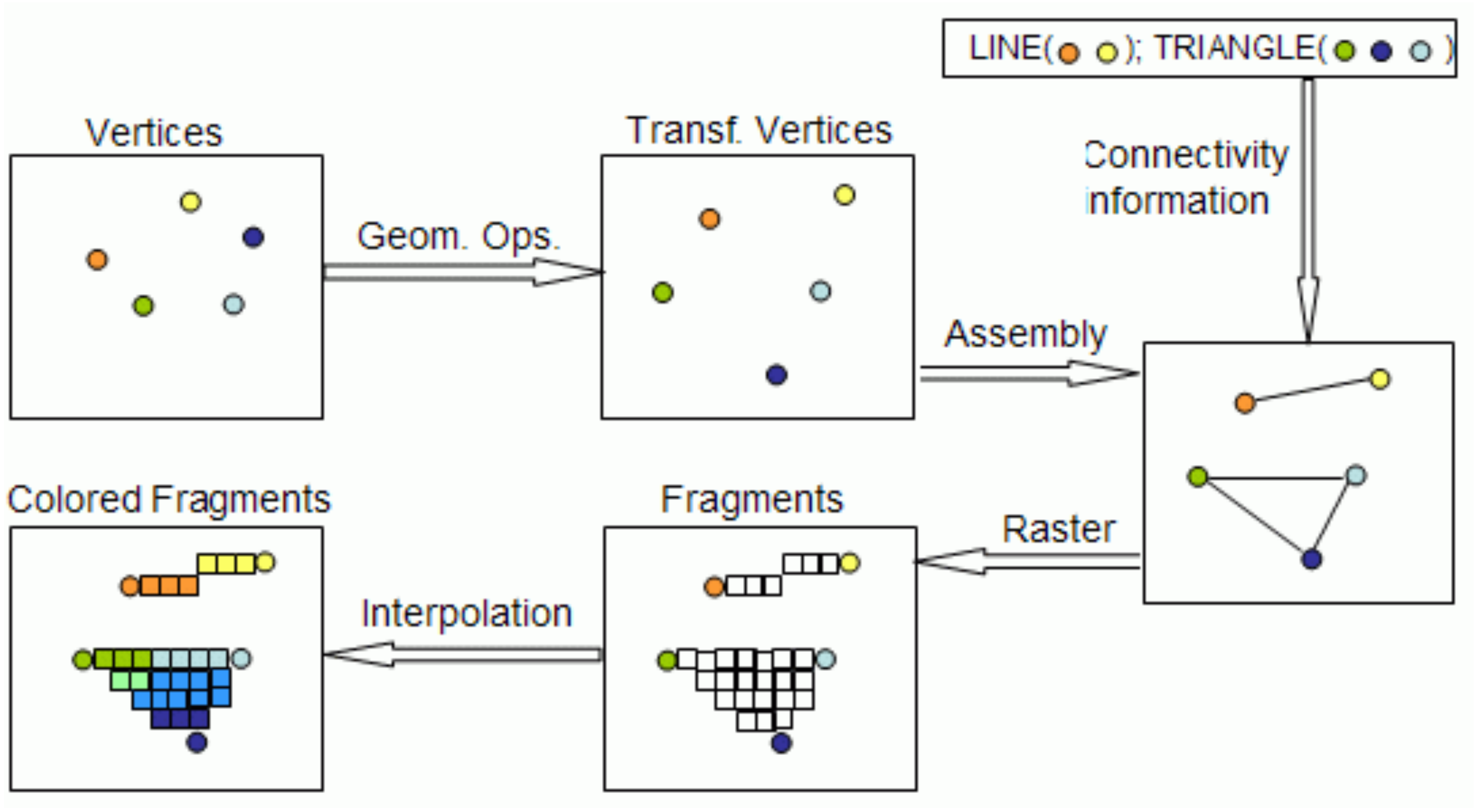
CSCI 173

California State University, Fresno

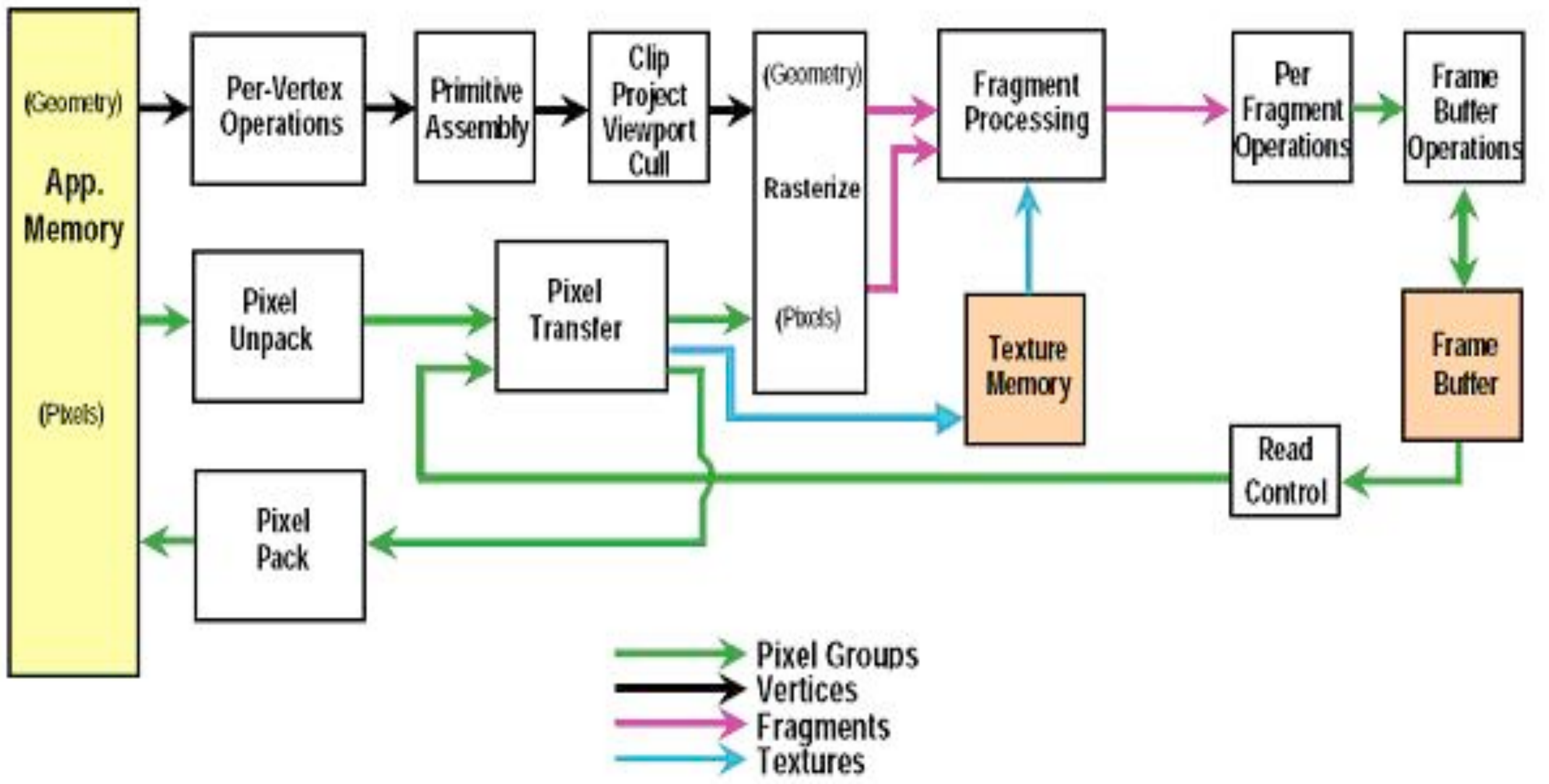
Shader PipeLINE overview



Visual Pipeline



Fixed function pipeline



Fixed v. Programmable

- Standard OpenGL: Fixed-function pipeline
- Add more user control & flexibility: programmable
- Pipeline processing - 2 stages
    - vertex processors
    - fragment processors

Vertex processor

- Vertex shader executed once for each vertex
- Vertex position transformation usually using the modelview and projection matrices
- Normal transformation, normalization
- Texture coordinate generation and transformation
- Lighting per vertex
- Color computation

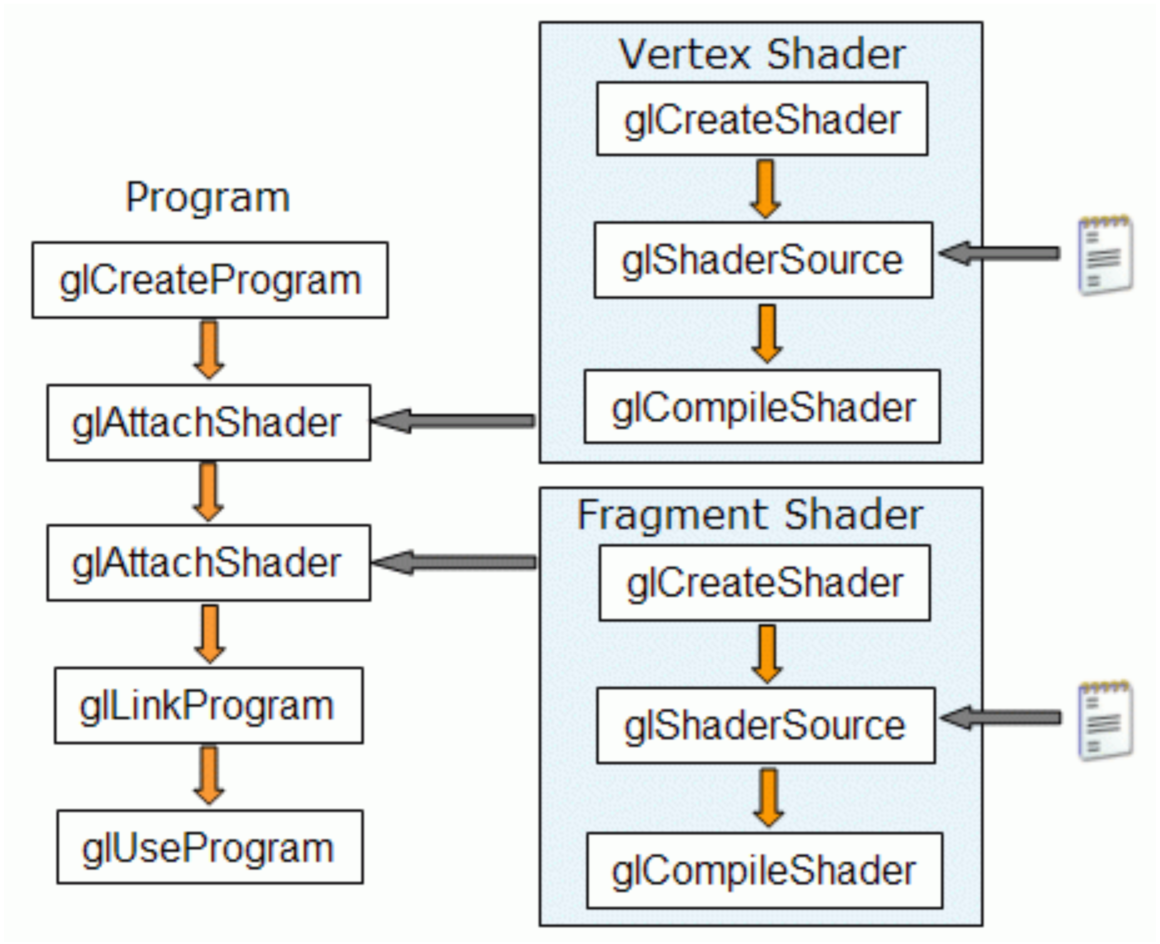
Fragment processor

- Fragment - per pixel data
- Fragment shader executed once for each fragment
- Computing colors and texture coordinates per pixel
- Texture application
- Fog computation
- Computing normals for lighting per pixel
- Can discard the fragment or compute color

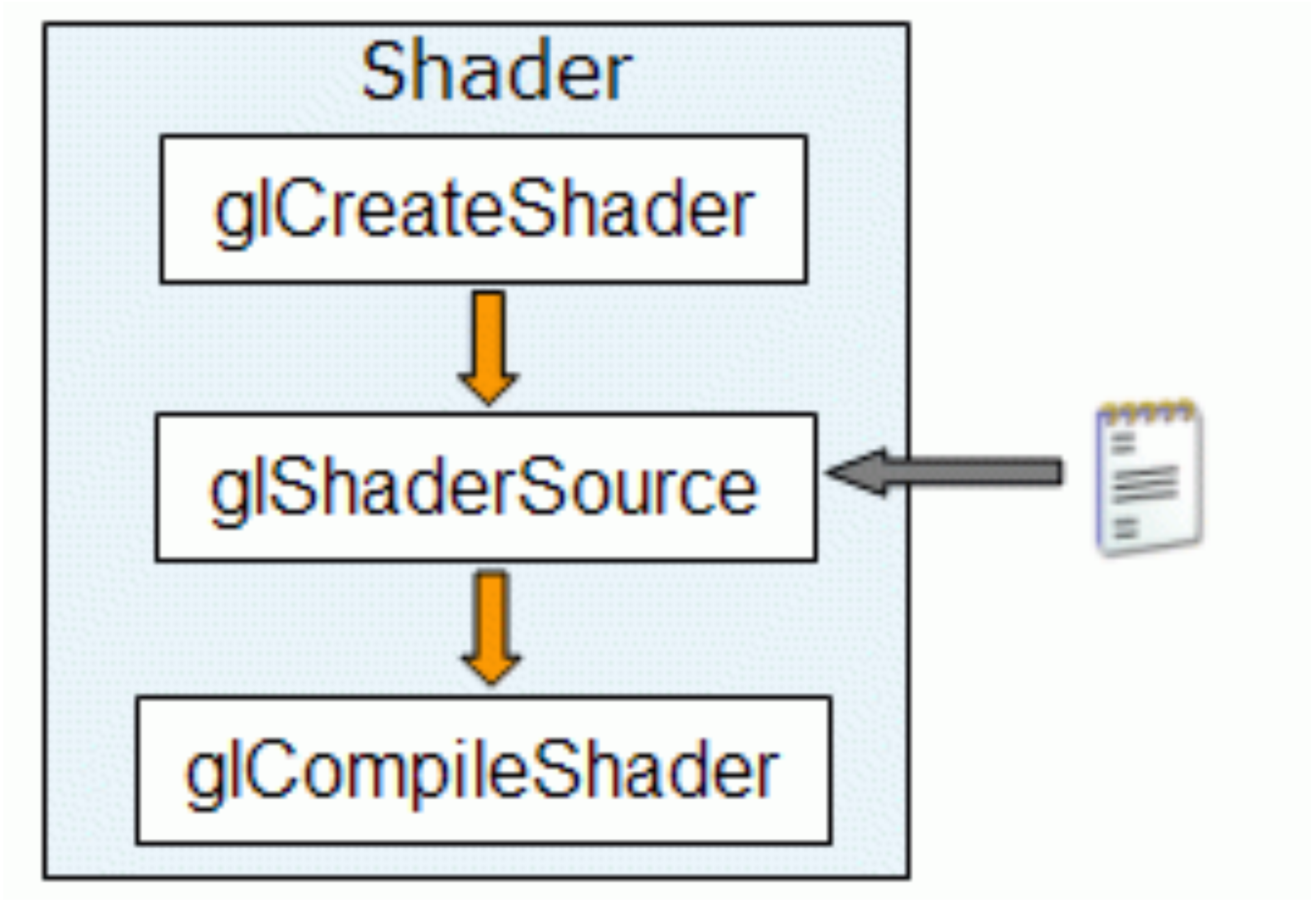
Setup for GLSL

- Each shader like a C module
  - compiled separately
  - linked to OpenGL program

Process Overview



Creating SHADERs

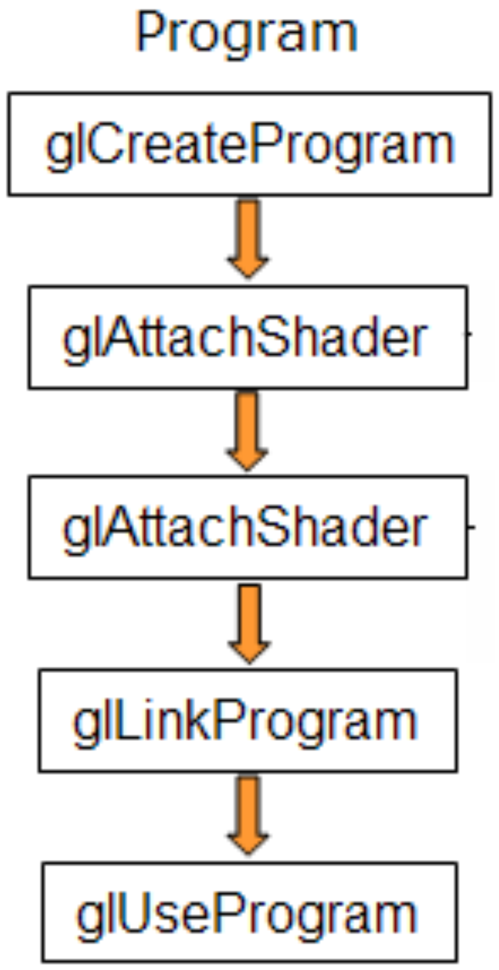


Process Overview

```
void glShaderSource(GLuint shader, int numOfStrings, const char **strings, int *lenOfStrings);
```

- shader – the handler to the shader.
- numOfStrings – the number of strings in the array.
- strings – the array of strings.
- lenOfStrings – an array with the length of each string, or NULL, meaning that the strings are NULL terminated.

Incorporating shaders



```
void setShaders() {

         char *vs,*fs;

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

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

         const char * vv = vs;
         const char * ff = fs;

         glShaderSource(v, 1, &vv,NULL);
         glShaderSource(f, 1, &ff,NULL);

         free(vs);free(fs);

         glCompileShader(v);
         glCompileShader(f);

         p = glCreateProgram();

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

         glLinkProgram(p);
         glUseProgram(p);
}
```

Debugging

Is hard
- no printf
- functions to test compilation & linking

e.g.

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

Can fetch an ‘infoLog’ to get more about errors

GLSL Variables

Read-only in shader

- value set by program

Uniform

- defined for a primitive (outside glBegin-glEnd)
- not on a per Vertex basis

Attribute

- on a per Vertex basis - for vertex shaders