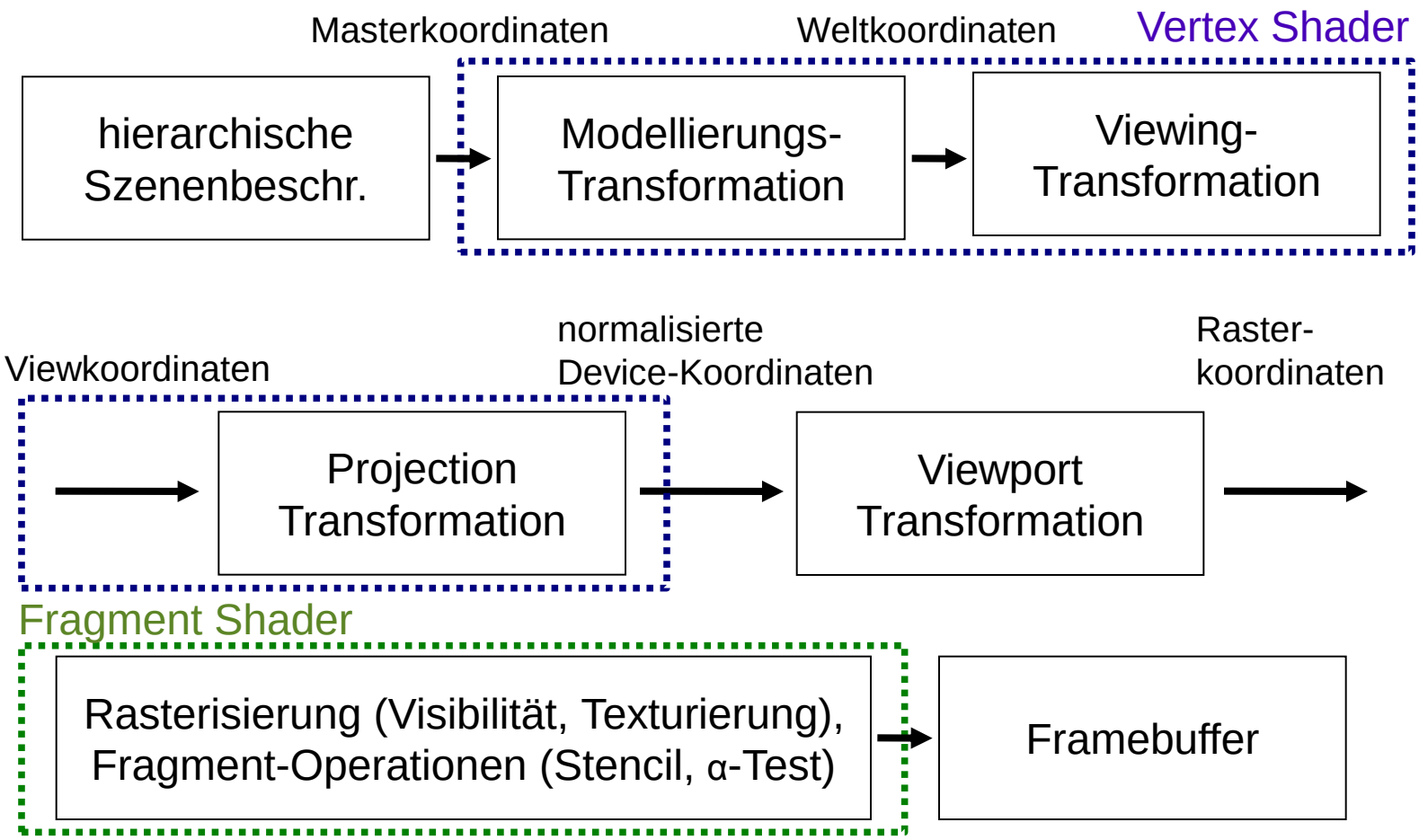
Graphik-Praktikum 2013

Aufgabe 2

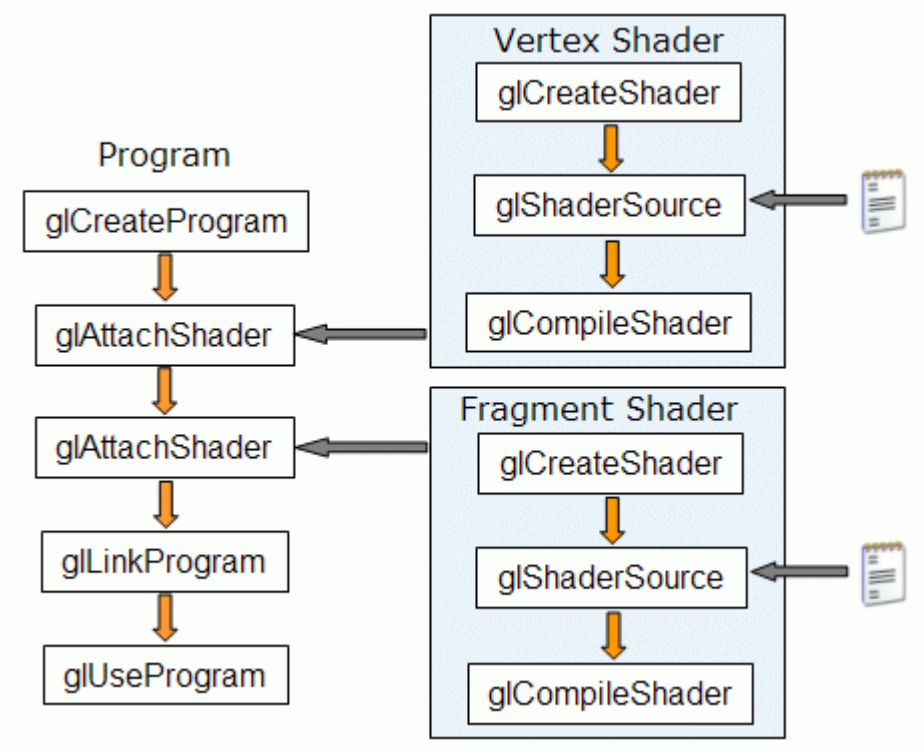
University of Siegen • Institute for Vision and Graphics
Computer Graphics and Multimedia Systems Group

16. Mai 2013

Pipeline Overview



Shader Kompilierung - OpenGL



- GLuint **glCreateShader**(GLenum shaderType);

  GL_VERTEX_SHADER_ARB or GL_FRAGMENT_SHADER_ARB

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

  lenOfStrings = NULL means NULL terminated strings

- void **glCompileShader**(GLuint shader);

- GLuint **glCreateProgram**(void);
- void **glAttachShader**(GLuint program, GLuint shader);
- void **glLinkProgram**(GLuint program);
- void **glUseProgram**(GLuint prog);

  prog = 0 bedeutet Fixed-Function-Pipeline

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

  type = GL_COMPILE_STATUS

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

  type = GL_LINK_STATUS

- void **glGetShaderInfoLog**(GLuint object, int maxLen, int *len, char *log);
- void **glGetProgramInfoLog**(GLuint object, int maxLen, int *len, char *log);
- Log length can be retrieved via previous functions using

  type = GL_INFO_LOG_LENGTH

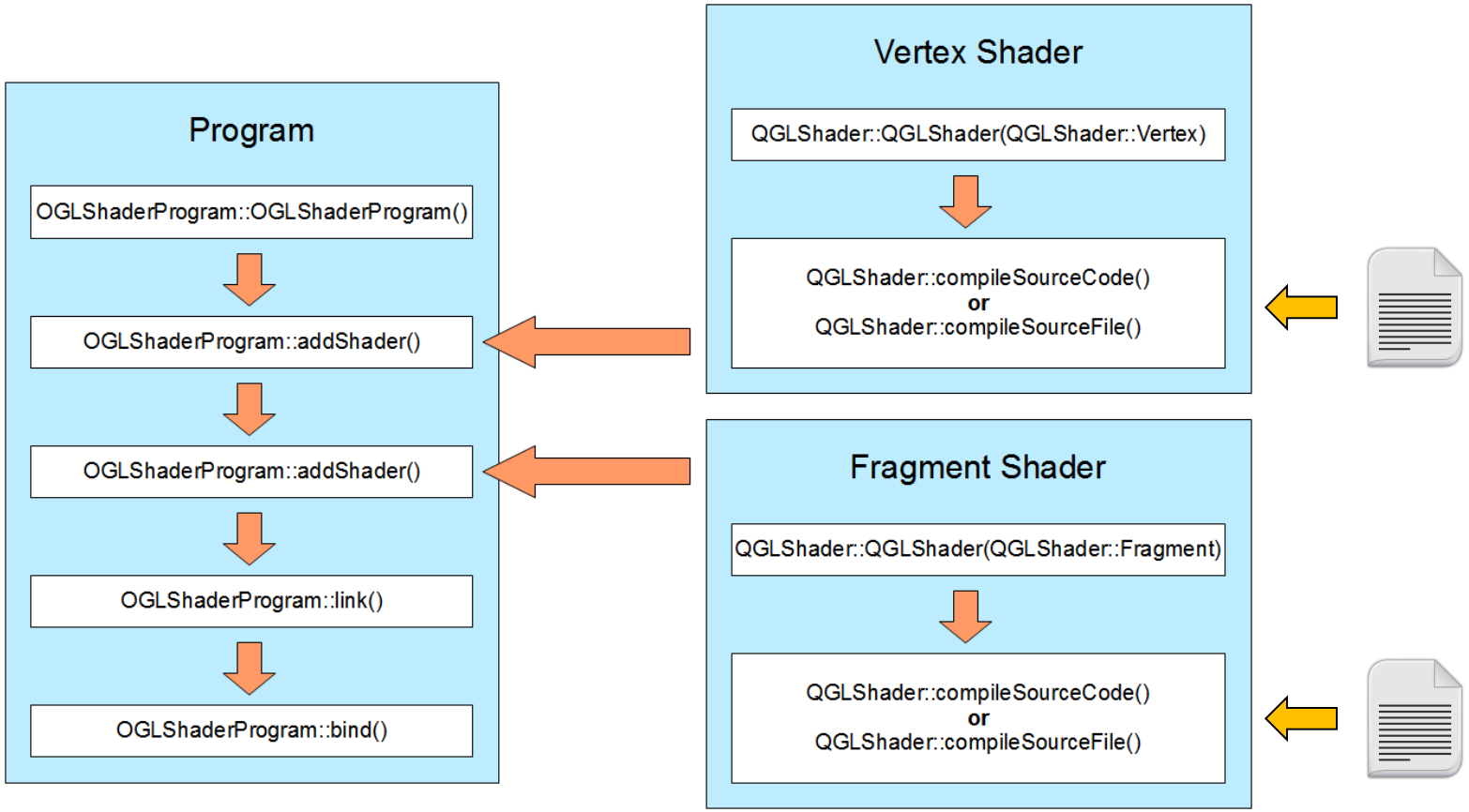
Cleanup

- void **glDetachShader**(GLuint program, GLuint shader);
- void **glDeleteShader**(GLuint id);
- void **glDeleteProgram**(GLuint id);

- GLint **glGetUniformLocation**(GLuint program, const char *name);
- void **glUniform{1,2,3,4}{i,f}**(GLint location, …);
- GLint **glUniformMatrix{2,3,4}fv**(GLint location, GLsizei count, GLboolean transpose, GLfloat *v);

Note, matrices are passed in column order!

Shader Kompilierung - QT



Shader Kompilierung - QT

- **QGLShader::QGLShader**(QGLShader::ShaderType, Qobject* parent = 0)

  QGLShader::Vertex, QGLShader::Fragment oder QGLShader::Geometry

  z.B.: *QGLShader* shader = new QGLShader(QGLShader::Vertex);*

- bool **QGLShader::compileSourceCode**(const QString& source)
- bool **QGLShader::compileSourceFile**(const QString& filename)

Shader-Programm Übersetzung - QT

- **QGLShaderProgram::QGLShaderProgram**(Qobject* parent = 0)

  QGLShaderProgram shaderProgram = new QGLShaderProgram();*

- bool **QGLShaderProgram::addShader**(QGLShader* shader)
- virtual bool **QGLShaderProgram::link**()
- bool **QGLShaderProgram::bind**()
- void **QGLShaderProgram::release**()

  schaltet zurück auf die Fixed-Function-Pipeline

- bool **QGLShader::isCompiled**()
- GLuint **QGLShader::shaderId**()

  hilfreich in Verbindung mit OpenGL Debugging-Funktionen

- bool **QGLShaderProgram::isLinked**()
- GLuint **QGLShaderProgram::programId**()

  hilfreich in Verbindung mit OpenGL Debugging-Funktionen

Cleanup - QT

- void **QGLShaderProgram::removeShader**(QGLShader* shader)
- void **QGLShaderProgram::removeAllShaders**()
- virtual **QGLShaderProgram::~QGLShaderProgram**()

  delete shaderProgram;

- virtual **QGLShader::~QGLShader**()

  delete shader;

Program Parameter - QT

- Hier nur Auszüge (mehr in der *QGLShaderProgram Class Reference*)
- void **QGLShaderProgram::setUniformValue**(const char* name, GL{float,int,uint} value)
- void **QGLShaderProgram::setUniformValue**(const char* name, const QVector{2,3,4}D value)
- void **QGLShaderProgram::setUniformValue**(const char* name, const Qmatrix{2,3,4}x{2,3,4} value)
- Anstelle von *const char* name* auch *int location* möglich

Data Types

- float, bool, int
- vec{2,3,4}, bvec{2,3,4}, ivec{2,3,4}
- mat{2,3,4}
- sampler{1,2,3}D
- uniform, varying
- in, out, inout

- **gl_ModelViewMatrix, gl_ProjectionMatrix**
  gl_ModelViewProjectionMatrix
- **gl_Vertex**
- **gl_Position =**
  gl_ProjectionMatrix * gl_ModelViewMatrix * gl_Vertex;
- vec4 **ftransform**(void);

  guarantees same result as fixed pipeline

- **gl_FragColor, gl_FragData[0]**
- **gl_Color, (gl_FrontColor)**

```
void main() {
    gl_FrontColor = gl_Color;
    gl_Position = ftransform();
    }
void main() {
    gl_FragColor = gl_Color;
    }
```

- gl_MultiTexCoord**X**
- gl_TexCoord[0] = gl_MultiTexCoord0;
- uniform sampler2D tex;
- vec4 **texture2D**(sampler2D, vec2);

```
void main() {
    gl_TexCoord[0] = gl_TextureMatrix[0] * gl_MultiTexCoord0;
    gl_Position = ftransform();
    }

uniform sampler2D tex;

void main() {
    vec4 color = texture2D(tex,gl_TexCoord[0].st);
    gl_FragColor = color;
    }
```

Statements

- if, then, else
- for (; ;), while () do, continue, break
- discard, vec4 dest = src.zxyw
- void main()
- **no recursion**

→ GLSL Quick Reference Guide, R. Rost „The OpenGL Shading Language“