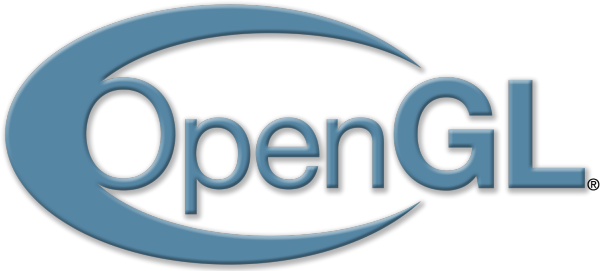
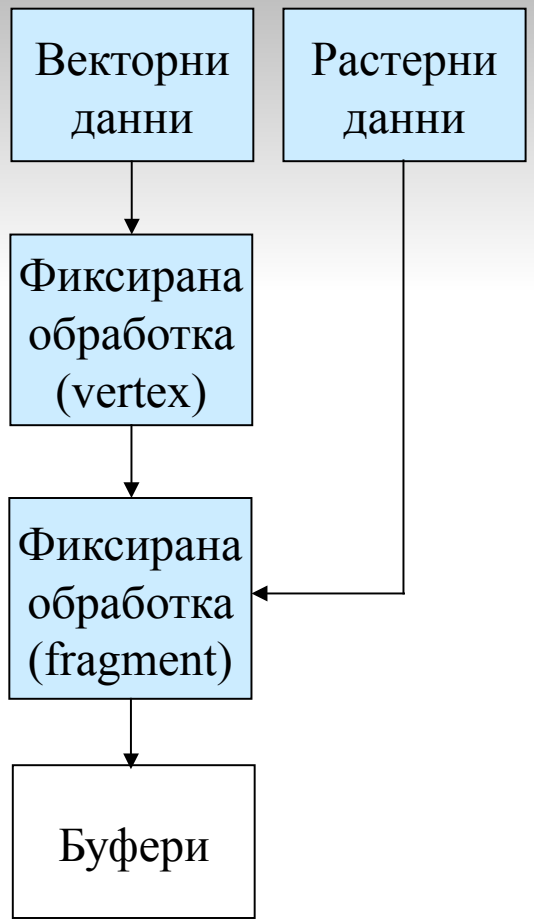
OpenGL

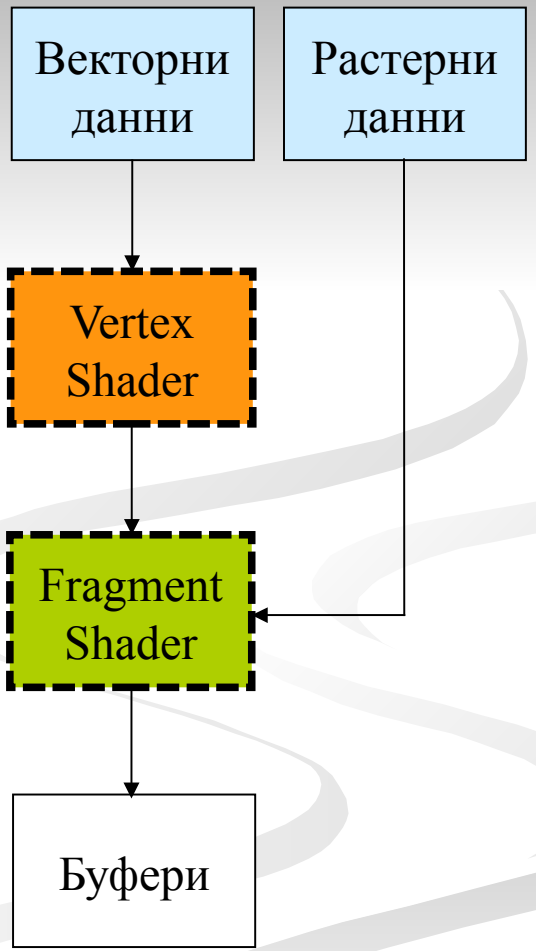
Shaders



гл. ас. д-р А. Пенев

Фиксиран или Програмируем





Някои GPU/драйвери не поддържат едновременна работа на фиксирана и програмируема функционалност.

Deprecation модел

- В OpenGL 3.1 “фиксираните” финкции са премахнати чрез препоръка за забрана за използване (deprecation);
- Разширението GL_ARB_compatibility;
- Голяма част от функциите на OpenGL трябва да се реализират в Shader-ите.

Пример (GLSL 1.30)

```
#version 130

uniform float t;    // Време. Предава се от програмата
vec4 vel;           // Скорост. Предава се от програмата

const vec4 g = vec4(0.0, -9.80, 0.0);

void main() {
  vec4 position = gl_Vertex;
  position += t*vel + t*t*g;
  gl_Position = gl_ModelViewProjectionMatrix * position;
}
```

Пример (GLSL 1.40)

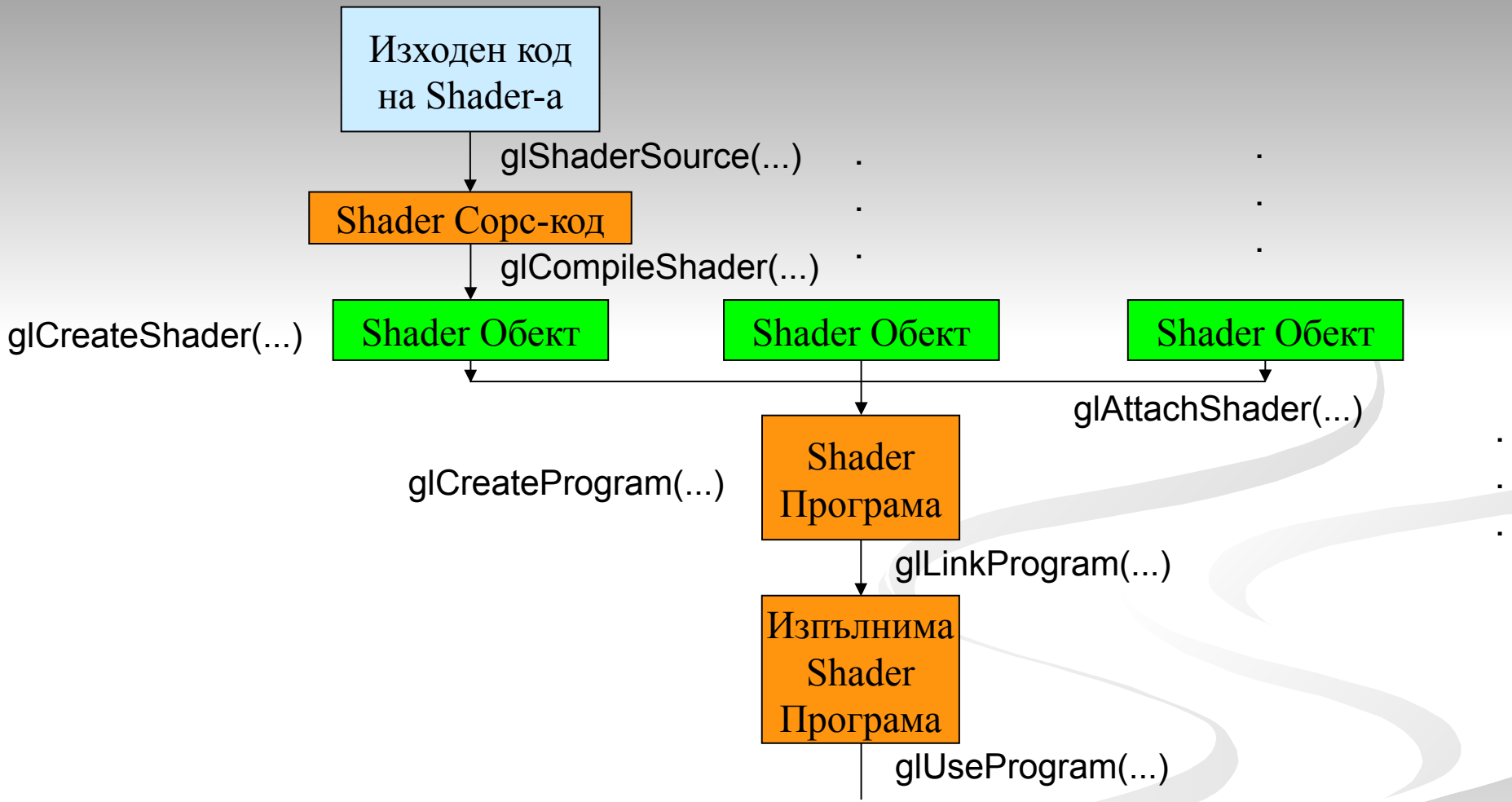
```
#version 140

uniform float t;    // Време. Предава се от програмата
uniform mat4 MVP;   // Комбинирана трансформация
in vec4 pos;        // Позиция
in vec4 vel;        // Скорост

const vec4 g = vec4(0.0, -9.80, 0.0);

void main() {
  vec4 position = pos;
  position += t*vel + t*t*g;
  gl_Position = MVP * position;
}
```

Схема на работа



Създаване на Shader обект

```
GLuint glCreateShader(GLenum type)
```

type:

GL_VERTEX_SHADER	Vertex shader
GL_FRAGMENT_SHADER	Fragment sh-r
GL_COMPUTE_SHADER	
GL_TESS_CONTROL_SHADER	
GL_TESS_EVALUATION_SHADER	
GL_GEOMETRY_SHADER	

Асоцииране на сорс-кода към Shader обекта

```
void glShaderSource(GLuint shader,
  GLsizei count,
  const GLchar **string,
  const GLint *length)
```

Ако length е NULL всеки низ се смята за NULL-терминиран

Компилиране на Shader обекта

```
void glCompileShader(GLuint shader)
```

Проверявайте за грешка!

`glGetShaderiv(...); glGetShaderInfoLog(...)`

Създаване на Shader програма

```
GLuint glCreateProgram()

void glAttachShader(GLuint program,
  GLuint shader)
void glDetachShader(GLuint program,
  GLuint shader)
```

Линкване на Shader програма

```
void glLinkProgram(GLuint program)
```

Проверявайте за грешки!

```
glGetProgramiv(...), glGetProgramInfoLog(...)
```

Параметри на Shader-ите

```
GLint glGetUniformLocation(
  GLuint program,
  const GLchar *name)
```

Използване на Shader програма

```
void glUseProgram(GLuint program)
```

Работа с параметрите на Shader-ите

```
void glUniform...(GLint location,
  GLtype v0, GLtype v1, ...)
void glUniformMatrix...(
  GLint location, GLsizei count,
  GLboolean transpose,
  const GLtype *value)

void glGetUniform...(GLuint prog,
  GLint location, GLtype *params)
```

Други

```
void glDeleteShader(GLuint shader);
void glDeleteProgram(GLuint prog);
GLboolean glIsProgram(GLuint prog);
GLboolean glIsShader(GLuint shader)
```

Пример (1/5)

```
const GLchar* vertexShaderSrc =
  "#version 150\n"
  "uniform mat4 viewMatrix, projMatrix;\n"
  "in vec4 position;\n"
  "in vec3 color;\n"
  "out vec3 Color;\n"
  "void main() {\n"
  "  Color = color;\n"
  "  gl_Position = projMatrix * viewMatrix * position;\n"
  "}\n";
```

Пример (2/5)

```
const GLchar* fragmentShaderSrc[] = {
  "#version 150",
  "in vec3 Color;",
  "out vec4 OutputFrag;",
  "void main() {",
  "  OutputFrag = vec4(Color, 1.0);",
  "}"
};
```

Пример (3/5)

```
void CheckCompileError(GLuint shader) {
  GLboolean status; GLint length; GLchar* log;
  glGetShaderiv(shader, GL_COMPILE_STATUS, &status);
  if (status == GL_TRUE) return;
  glGetShaderiv(shader, GL_INFO_LOG_LENGTH, &length);
  log = (GLchar*) malloc(length);
  glGetShaderInfoLog(shader, length, &length, log);
  fprintf(stderr, "Compile log = '%s'\n", log);
  free(log);
  exit(EXIT_FAILURE);
}

void CheckLinkError(GLuint program) {
  GLboolean status; GLint length; GLchar* log;
  glGetProgramiv(program, GL_LINK_STATUS, &status);
  if (status == GL_TRUE) return;
  glGetProgramiv(program, GL_INFO_LOG_LENGTH, &length);
  log = (GLchar*) malloc(length);
  glGetProgramInfoLog(program, length, &length, log);
  fprintf(stderr, "Link log = '%s'\n", log);
  free(log);
  exit(EXIT_FAILURE);
}
```

Пример (4/5)

```
// Global
GLuint vertexLoc, colorLoc;
GLuint projMatrixLoc, viewMatrixLoc;

// B main
GLuint vertexShader = glCreateShader(GL_VERTEX_SHADER);
glShaderSource(vertexShader, 1, &vertexShaderSrc, NULL);
glCompileShader(vertexShader);
CheckCompileError(vertexShader);

GLuint fragmentShader = glCreateShader(GL_FRAGMENT_SHADER);
glShaderSource(fragmentShader, 6, fragmentShaderSrc, NULL);
glCompileShader(fragmentShader);
CheckCompileError(fragmentShader);

GLuint program = glCreateProgram();
glAttachShader(program, vertexShader);
glAttachShader(program, fragmentShader);
glBindFragDataLocation(program, 0, "OutputFrag");

glLinkProgram(program);
CheckLinkError(program);
```

Пример (5/5)

```
vertexLoc = glGetAttribLocation(p, "position");
colorLoc = glGetAttribLocation(p, "color");
projMatrixLoc = glGetUniformLocation(p, "projMatrix");
viewMatrixLoc = glGetUniformLocation(p, "viewMatrix");
```

```
// В display

...

glUseProgram(program);

glUniformMatrix4fv(projMatrixLoc, 1, false, projMatrix);
glUniformMatrix4fv(viewMatrixLoc, 1, false, viewMatrix);

...

// Визуализация на геометрията

...
```

OpenGL Shading Language (GLSL)

- **GLSL е *“C”*** подобен език;
- Има проста семантика подобна на ***“C”***;
- В него се използват някои специфични типове данни и функции.

GLSL модификатори

```
const       // константи
in          // входна за shader-а
out         // изходна
uniform     // конст. за примитива
```

И много други като: buffer, shared, flat, smooth, noperspective, sample, readonly, writeonly, coherent, volatile, restrict, invariant, ~~attribute~~, ~~varring~~, ...

GLSL типове (1/3)

```
int                 // цяло число
uint                // цяло без знак
bool                // логически
float               // реално число
double              // реално число
                    // с двойна
                    // точност
```

Типа double се поддържа само в OpenGL 4.0 или ако е налично ARB_gpu_shader_fp64
В shader-а трябва да се добави и `#define GL_ARB_gpu_shader_fp64 1`

GLSL типове (2/3)

```
vec2, vec3, vec4    // float вектори
```

Аналогично има (n=2..4):

```
bvecn               // булев вектор
ivecn               // целочислен
uvecn               // цял без знак
dvecn               // double вектор
```

GLSL типове (3/3)

```
mat2, mat3, mat4          // матрици
mat2x2, mat3x3, mat4x4
mat3x2, mat3x3, mat3x4
mat4x2, mat4x3, mat4x4
```

Аналогично има типове за матрици с реални числа с двойна точност:

```
dmat...
```

GLSL инициализация на променливи и константи

```
int i = 0;
vec3 v = vec3(1.0, 0.0, 0.0);
vec4 v4 = vec4(1.0, 0.0, 0.0, 1.0);
vec4 v4a = vec4(v, 1.0);
mat3 M = mat3(1.0, 2.0, 3.0,
              4.0, 5.0, 6.0,
              7.0, 8.0, 9.0);
mat4 M4 = mat4(4.0);
```

GLSL достъп до елементите

```
vec3 v = vec3(1.0, 2.0, 3.0);
vec4 c = vec4(1.0, 2.0, 3.0, 4.0);

v.x
c.r
c.rgba
c.rrr
...
```

Използват се *xyzw* за вектори, *rgba* за цветове и *stpq* за текстурни координати.

GLSL структури

```
struct Particle {
   float lifetime;
   vec3 position;
   vec3 velocity;
};

Particle p = Particle(1.0,pos,vel);
```

GLSL масиви

```
int indices[];

float coef[3] =
   float[3](1.0, 2.0, 3.0);

for (int i=0; i<coef.length(); i++)
{
  coef[i] *= 2.0;
}
```

GLSL предефинирани оператори

```
vec4 v1 = vec4(1.0, 2.0, 3.0, 4.0);
vec4 v2 = vec4(5.0, 6.0, 7.0, 8.0);
mat4 M = mat4(2.0);

float a = v1 * v2
vec4 v3 = M * v1;
```

OpenGL Shading Language

- Цикли (for, while, do-while);
- Условни оператори (if-else, switch-case);
- Изрази;
- Потребителски функции (без рекурсивни);
- ...

OpenGL - Shaders

Въпроси?