



**LOBACHEVSKY STATE UNIVERSITY**  
**of NIZHNI NOVGOROD**  
National Research University

Computing Mathematics and Cybernetics faculty  
Software department

Computer Graphics. Introduction Course

Обработка изображения на  
шейдерах. Простейшая

программа.

Белокаменская А.А., Васильев Е.П.

# Шаг 1

---

- За основу берем приложение OpenGL из предыдущей темы.
- Для того, чтобы применить фильтр к картинке используем текстуру, в которую загрузим исходную картинку. Копируем функцию загружающую текстуру из файла.
- Эту текстуру передадим во фрагментный шейдер, где для каждого пиксела рассчитаем цвет, учитывая влияние фильтра.
- Вершинный шейдер только преобразует координаты квада, на котором будем отображать результаты расчетов

# Вершинный шейдер

---

- Шейдеры – это два текстовых файла. Их нужно загрузить с диска и скомпилировать в шейдерную программу.
- Создадим два пустых текстовых файла «SepiaShader.vs» для вершинного шейдера и «SepiaShader.fs» – для фрагментного.
- Вершинный шейдер совсем короткий:

```
varying vec2 TexCoord; // Текстурные координаты
```

```
void main(void)  
{
```

```
    // Сохраняем текстурные координаты
```

```
    TexCoord = gl_MultiTexCoord0.st;
```

```
    // Вычисляем положение вершины в пространстве отсечения
```

```
    gl_Position = gl_ModelViewProjectionMatrix * gl_Vertex;
```

```
}
```

# Фрагментный шейдер

---

```
uniform sampler2D RenderTexture;           // Текстура с исходным изображением

varying vec2 TexCoord;                     // Текстурные координаты

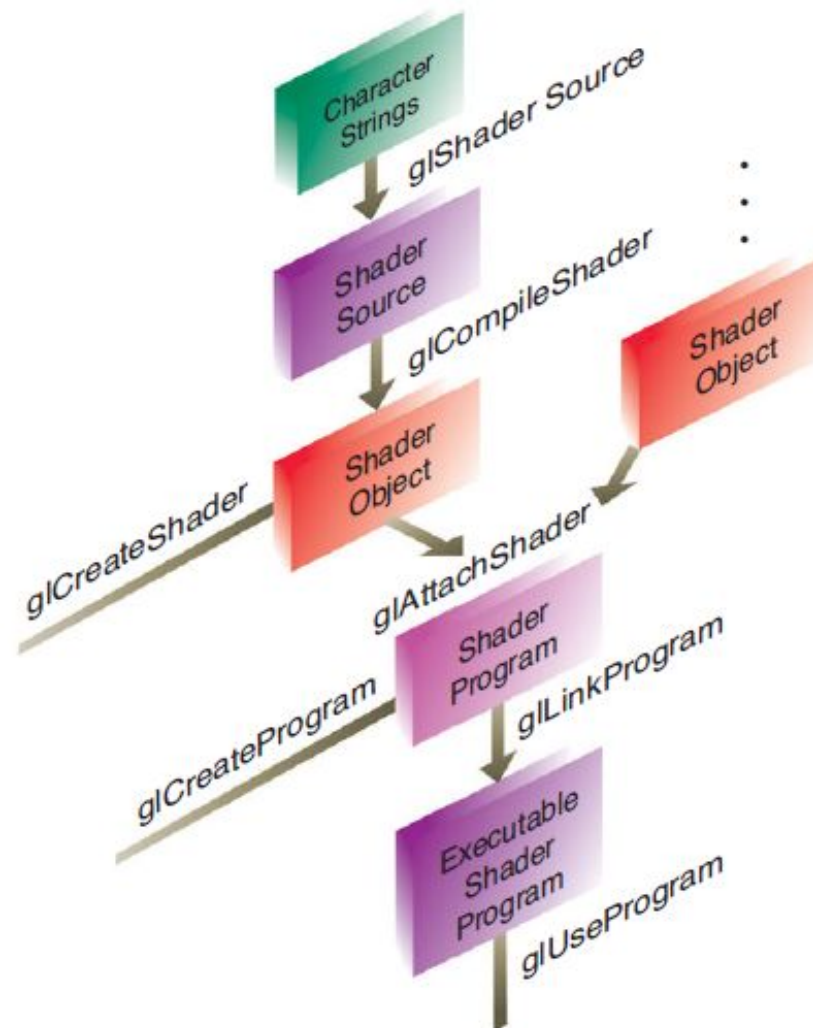
const vec3 Luminance = vec3(0.3, 0.59, 0.11); // Коэффициенты интенсивности компонент цвета

const vec3 Sepia = vec3(1.0, 0.89, 0.54);  // Цвет фильтра 'Sepia'

void main(void)
{
    // Получаем исходный цвет фрагмента
    vec3 color = vec3(texture(RenderTexture, TexCoord));

    // Вычисляем новый цвет фрагмента
    gl_FragColor = vec4(dot(color, Luminance) * Sepia, 1.0);
}
```

# Инициализация шейдерной программы



# Загрузка шейдеров

---

```
void loadShader(String filename, ShaderType type, int program, out int address)
{
    address = GL.CreateShader(type);
    using (System.IO.StreamReader sr = new StreamReader(filename))
    {
        GL.ShaderSource(address, sr.ReadToEnd());
    }
    GL.CompileShader(address);
    GL.AttachShader(program, address);
    Console.WriteLine(GL.GetShaderInfoLog(address));
}
```

# Инициализация шейдерной программы

---

```
int SepiaProgramID;
int SepiaVertexShader;
int SepiaFragmentShader;
int uniformRenderTexture;
int imageTextureID;

private bool InitShaders()
{
    SepiaProgramID = GL.CreateProgram();
    loadShader("../..\\Shaders\\SepiaShader.vs", ShaderType.VertexShader, SepiaProgramID, out SepiaVertexShader);
    loadShader("../..\\Shaders\\SepiaShader.fs", ShaderType.FragmentShader, SepiaProgramID, out SepiaFragmentShader);
    GL.LinkProgram(SepiaProgramID);
    Console.WriteLine(GL.GetProgramInfoLog(SepiaProgramID));
    GL.Enable(EnableCap.Texture2D);
    imageTextureID = CreateTexture("../..\\Images\\boat.jpg");
    uniformRenderTexture = GL.GetUniformLocation(SepiaProgramID, "RenderTexture");
    GL.Uniform1(uniformRenderTexture, imageTextureID);
    return true;
}
```

# Последние штрихи

---

- Добавляем вызов `InitShaders()` после `Init()`
- Разрешаем использование 2D текстур
- Вызываем `GL.UseProgram(SepiaProgramID)`; перед отрисовкой квада
- Добавляем генерацию текстурных координат квада и назначаем загруженную текстуру текущей перед отрисовкой.
- Запускаем.