Shaders

Nossa própria classe de sombreador

|  |
| --- |
| namespace LearnOpenTK.src {  internal class Shader {  }  } |

|  |
| --- |
| namespace LearnOpenTK.src {  internal class Shader {  public int ID;  public Shader(string vertexPath, string fragmentPath) {  }  public void use() {  }  public void setBool(string name, bool value) {  }  public void setInt(string name, int value) {  }  public void setFloat(string name, float value) {  }  }  } |

|  |
| --- |
| public Shader(string vertexPath, string fragmentPath) {  string vertexCode = File.ReadAllText("../../../src/Shaders/" + vertexPath);  string fragmentCode = File.ReadAllText("../../../src/Shaders/" + fragmentPath);  int vertex;  int fragment;  int success;  string infoLog;  // vertex Shader  vertex = GL.CreateShader(ShaderType.VertexShader);  GL.ShaderSource(vertex, vertexCode);  GL.CompileShader(vertex);  GL.GetShader(vertex, ShaderParameter.CompileStatus, out success);  if(success == 0) {  infoLog = GL.GetShaderInfoLog(vertex);  Console.WriteLine("ERROR::SHADER::VERTEX::COMPILATION\_FAILED\n" + infoLog);  }  // fragment Shader  fragment = GL.CreateShader(ShaderType.FragmentShader);  GL.ShaderSource(fragment, fragmentCode);  GL.CompileShader(fragment);  GL.GetShader(fragment, ShaderParameter.CompileStatus, out success);  if(success == 0) {  infoLog = GL.GetShaderInfoLog(fragment);  Console.WriteLine("ERROR::SHADER::VERTEX::COMPILATION\_FAILED\n" + infoLog);  }  // shader Program  ID = GL.CreateProgram();  GL.AttachShader(ID, vertex);  GL.AttachShader(ID, fragment);  GL.LinkProgram(ID);  GL.GetProgram(ID, GetProgramParameterName.LinkStatus, out success);  if(success == 0) {  infoLog = GL.GetProgramInfoLog(ID);  Console.WriteLine("ERROR::SHADER::PROGRAM::LINKING\_FAILED\n" + infoLog);  }  GL.DeleteShader(vertex);  GL.DeleteShader(fragment);  } |

|  |
| --- |
| public void use() {  GL.UseProgram(ID);  } |

|  |
| --- |
| public void setBool(string name, bool value) {  GL.Uniform1(GL.GetUniformLocation(ID, name), value ? 1 : 0);  }  public void setInt(string name, int value) {  GL.Uniform1(GL.GetUniformLocation(ID, name), value);  }  public void setFloat(string name, float value) {  GL.Uniform1(GL.GetUniformLocation(ID, name), value);  } |

|  |
| --- |
| Shader ourShader; |

OnLoad()

|  |
| --- |
| ourShader = new Shader("shader.vert", "shader.frag"); |

OnRenderFrame()

|  |
| --- |
| ourShader.use(); |