Teaching a Modern Graphics Pipeline Using a Shader-based Software Renderer

Heinrich Fink
Thomas Weber
Michael Wimmer

1Institute of Computer Graphics and Algorithms, Vienna University of Technology
Our Introductory CG Lab Course
Understand & implement a software renderer
Our Introductory CG Lab Course
Scene Outliner
- portrait
- backlight
- fillight
- keylight

Croctopus
Geometry
Transform
Matrix
0.76  0.65  0.00  0.00
-0.65  0.76  0.00  0.00
0.00  0.00  1.00  0.00
0.00  0.00  0.00  1.00

Reset
Identity

Apply Transformations
Multiply from: Left

Translate

Rotate
X:
Y:
Z:

Scale

Reset matrix to initial state (at the time of loading).
Test & explore renderer features within 3D editor

Reset matrix to initial state (at the time of loading).
Reset matrix to initial state (at the time of loading).
Our Introductory CG Lab Course

- Only requires basic Java programming skills
- No prior experience in computer graphics necessary
- 3 ECTS lab course (~75 hrs per semester)
Our Introductory CG Lab Course

- Only requires basic Java programming skills
- No prior experience in computer graphics necessary
- 3 ECTS lab course (~75 hrs per semester)
Our Introductory CG Lab Course

- Only requires basic Java programming skills
- No prior experience in computer graphics necessary
- 3 ECTS lab course (~75 hrs per semester)

- We provide an initial software renderer framework
Our Introductory CG Lab Course

- Only requires basic Java programming skills
- No prior experience in computer graphics necessary
- 3 ECTS lab course (~75 hrs per semester)

- We provide an initial software renderer framework
- Students complete six programming assignments
Our Introductory CG Lab Course

- Only requires basic Java programming skills
- No prior experience in computer graphics necessary
- 3 ECTS lab course (~75 hrs per semester)

- We provide an initial software renderer framework
- Students complete six programming assignments
- Assignments complete key features of the software renderer and build upon each other
Our Introductory CG Lab Course

- Only requires basic Java programming skills
- No prior experience in computer graphics necessary
- 3 ECTS lab course (~75 hrs per semester)

- We provide an initial software renderer framework
- Students complete six programming assignments
- Assignments complete key features of the software renderer and build upon each other
- A rich 3D editor is contained within the framework and allows to experiment with the implemented solutions
Assignment 1
Line Rasterization
Rendered directly by student’s solution
Rendered directly by student’s solution
Rendered directly by student’s solution
+ Line rasterization
Select a camera navigation mode from the top.
Viewport mapping
Assignment 2
Transformations in 3D Space
Vec3.java
Vec4.java
Mat3.java
Mat4.java

+ 3D math

Vec3.java
Vec4.java
Mat3.java
Mat4.java
+ Model transformations
Assignment 3
Clipping, Viewing & Projection
+ Projection Transform

PerspectiveProjection.java
Renderer.java
ColorShader.java
Assignment 4
Triangle Rasterization & Culling
Click fields to edit the transformation matrix. Right-click drag to continuously change values.
+ Back-face culling

Click fields to edit the transformation matrix. Right-click drag to continuously change values.
Click fields to edit the transformation matrix. Right-click drag to continuously change values.
Assignment 5
Lighting & Shading
Click to select, double-click to expand nodes, triple-click to rename objects. Use arrow keys to navigate.
+ Per-vertex diffuse reflection

IlluminationModels.java
PointLight.java
Geometry.java
LambertGouraudShader.java
Scene.java
Click to select, double-click to expand nodes, triple-click to rename objects. Use arrow keys to navigate.
Per-fragment specular reflection

IlluminationModels.java
BlinnPhongShader.java
Scene.java
nd nodes, triple-click to rename objects. Use arrow keys to navigate.
Assignment 6
Texturing & Custom Shaders
public class TexturedShader implements Shader {

    @Image2DSamplerParam(name="Diffuse Sampler")
    public Sampler diffuse;

    @RGBParam(r=1, g=1, b=1)
    public Vec3 specular;

    @FloatParam(value=20, min=1, max=200.0f)
    public float shininess;

    @Override
    public Vertex shadeVertex(Mesh.Vertex v) {
        ...
    }

    @Override
    public Vec3 shadeFragment(Varyings varyings) {
        ...
    }
}
public class TexturedShader implements Shader {

    @Image2DSamplerParam(name="Diffuse Sampler")
    public Sampler diffuse;

    @RGBParam(r=1, g=1, b=1)
    public Vec3 specular;

    @FloatParam(value=20, min=1, max=200.0f)
    public float shininess;

    @Override
    public Vertex shadeVertex(Mesh.Vertex v) {
        ...
    }

    @Override
    public Vec3 shadeFragment(Varyings varyings) {
        ...
    }
}
public class TexturedShader implements Shader {

    @Image2DSamplerParam(name="Diffuse Sampler")
    public Sampler diffuse;

    @RGBParam(r=1, g=1, b=1)
    public Vec3 specular;

    @FloatParam(value=20, min=1, max=200.0f)
    public float shininess;

    @Override
    public Vertex shadeVertex(Mesh.Vertex v)
    {...

    @Override
    public Vec3 shadeFragment(Varyings varyings)
    {...
}
public class TexturedShader implements Shader {

  @Image2DSamplerParam(name="Diffuse Sampler")
  public Sampler diffuse;

  @RGBParam(r=1, g=1, b=1)
  public Vec3 specular;

  @FloatParam(value=20, min=1, max=200.0f)
  public float shininess;

  @Override
  public Vertex shadeVertex(Mesh.Vertex v) {
...
  }

  @Override
  public Vec3 shadeFragment(Varyings varyings) {
...
  }
}
public class TexturedShader implements Shader {

  @Image2DSamplerParam(name="Diffuse Sampler")
  public Sampler diffuse;

  @RGBParam(r=1, g=1, b=1)
  public Vec3 specular;

  @FloatParam(value=20, min=1, max=200.0f)
  public float shininess;

  @Override
  public Vertex shadeVertex(Mesh.Vertex v) {
    ...
  }

  @Override
  public Vec3 shadeFragment(Varyings varyings) {
    ...
  }
}
public class TexturedShader implements Shader {

    @Image2DSamplerParam(name="Diffuse Sampler")
    public Sampler diffuse;

    @RGBParam(r=1, g=1, b=1)
    public Vec3 specular;

    @FloatParam(value=20, min=1, max=200.0f)
    public float shininess;

    @Override
    public Vertex shadeVertex(Mesh.Vertex v) {
        ...
    }

    @Override
    public Vec3 shadeFragment(Varyings varyings) {
        ...
    }
}
public class TexturedShader implements Shader {

  @Image2DSamplerParam(name="Diffuse Sampler")
  public Sampler diffuse;

  @RGBParam(r=1, g=1, b=1)
  public Vec3 specular;

  @FloatParam(value=20, min=1, max=200.0f)
  public float shininess;

  @Override
  public Vertex shadeVertex(Mesh.Vertex v)
  {
      ...
  }

  @Override
  public Vec3 shadeFragment(Varyings varyings)
  {
      ...
  }
}
public class TexturedShader implements Shader {

    @Image2DSamplerParam(name="Diffuse Sampler")
    public Sampler diffuse;

    @RGBParam(r=1, g=1, b=1)
    public Vec3 specular;

    @FloatParam(value=20, min=1, max=200.0f)
    public float shininess;

    @Override
    public Vertex shadeVertex(Mesh.Vertex v) {
        ...
    }

    @Override
    public Vec3 shadeFragment(Varyings varyings) {
        ...
    }
}
public class TexturedShader implements Shader {

  @Image2DSamplerParam(name="Diffuse Sampler")
  public Sampler diffuse;

  @RGBParam(r=1, g=1, b=1)
  public Vec3 specular;

  @FloatParam(value=20, min=1, max=200.0f)
  public float shininess;

  @Override
  public Vertex shadeVertex(Mesh.Vertex v) {
      ...
  }

  @Override
  public Vec3 shadeFragment(Varyings varyings) {
      ...
  }
}
+ Own Shader Creation

MyCustomShader.java

MyCustomShader.java

Friday, November 9, 12
Track Mode: Move the camera on its local XY-plane (Alt + MMB).
In our course undergraduate students implement fundamental features of a software renderer.

Our framework provides the initial code base and an interactive 3D editor around it.

Playful exploration of features and rich content interoperability highly motivates students.