UNIVERSITÄT
DES
SAARLANDES

Programming 2 - SS23

Project 5 - Raytracer

Authors: Aiman Al-Azazi, Fabian Waller, Maike Kalms

21. June 2023

Saarland University

Overview

1. Organization

2. Project Structure

3. Linear Algebra

4. Modeling

5. Light Sources

6. Running the Project

Organization

Git Project-Repository

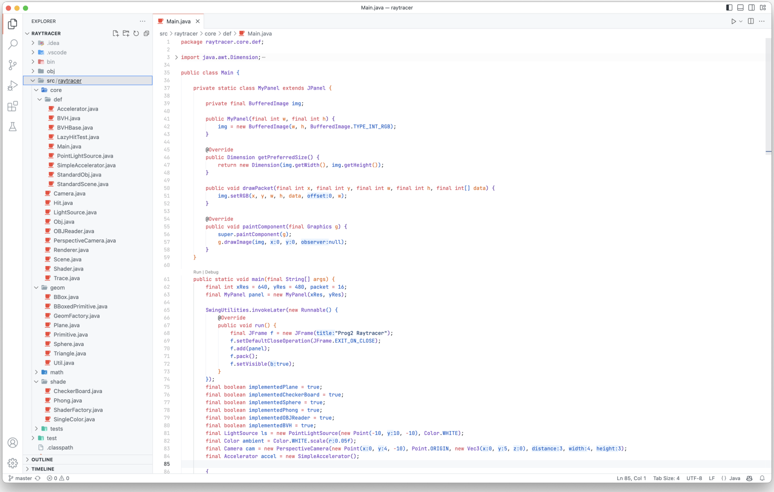
You can get the project files with `git clone` under the following url:

```
ssh://git@git.prog2.de:2222/project5/$NAME.git
```

$NAME = Your username in the CMS

Project Structure

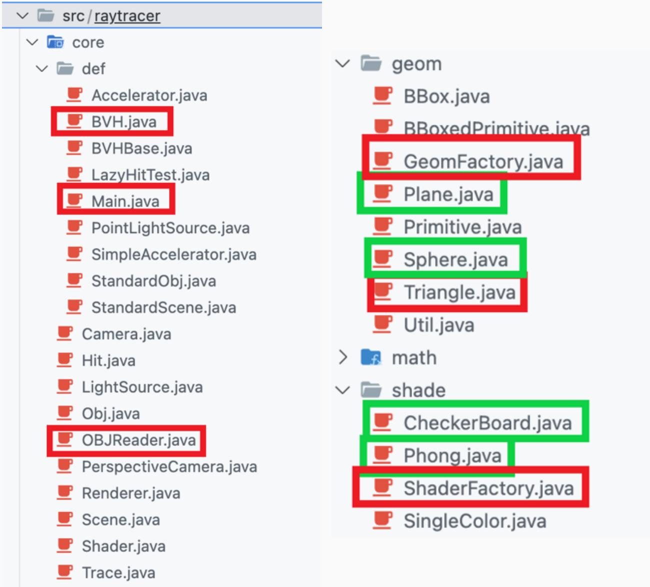
Overview



```
RAYTRACER
  .idea
  .vscode
  bin
  obj
  src / raytracer
    core
      def
        Accelerator.java
        BVH.java
        BVHBase.java
        LazyHitTest.java
        Main.java
        PointLightSource.java
        SimpleAccelerator.java
        StandardObj.java
        StandardScene.java
      Camera.java
      Hit.java
      LightSource.java
      Obj.java
      OBJReader.java
      PerspectiveCamera.java
      Renderer.java
      Scene.java
      Shader.java
      Trace.java
    geom
      BBox.java
      BBoxedPrimitive.java
      GeomFactory.java
      Plane.java
      Primitive.java
      Sphere.java
      Triangle.java
      Util.java
    math
    shade
      CheckerBoard.java
      Phong.java
      ShaderFactory.java
      SingleColor.java
    tests
  test
  .classpath
```

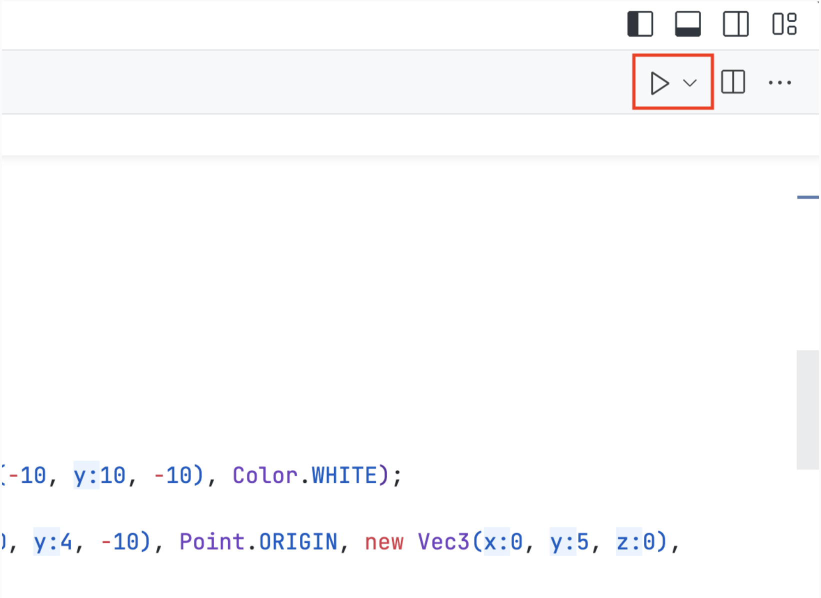
```java
package raytracer.core.def;

import java.awt.Dimension;…

public class Main {

    private static class MyPanel extends JPanel {

        private final BufferedImage img;

        public MyPanel(final int w, final int h) {
            img = new BufferedImage(w, h, BufferedImage.TYPE_INT_RGB);
        }

        @Override
        public Dimension getPreferredSize() {
            return new Dimension(img.getWidth(), img.getHeight());
        }

        public void drawPacket(final int x, final int y, final int w, final int h, final int[] data) {
            img.setRGB(x, y, w, h, data, offset:0, w);
        }

        @Override
        public void paintComponent(final Graphics g) {
            super.paintComponent(g);
            g.drawImage(img, x:0, y:0, observer:null);
        }
    }

    public static void main(final String[] args) {
        final int xRes = 640, yRes = 480, packet = 16;
        final MyPanel panel = new MyPanel(xRes, yRes);

        SwingUtilities.invokeLater(new Runnable() {
            @Override
            public void run() {
                final JFrame f = new JFrame(title:"Prog2 Raytracer");
                f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
                f.add(panel);
                f.pack();
                f.setVisible(b:true);
            }
        });
        final boolean implementedPlane = true;
        final boolean implementedCheckerBoard = true;
        final boolean implementedSphere = true;
        final boolean implementedPhong = true;
        final boolean implementedOBJReader = true;
        final boolean implementedBVH = true;
        final LightSource ls = new PointLightSource(new Point(-10, y:10, -10), Color.WHITE);
        final Color ambient = Color.WHITE.scale(r:0.05f);
        final Camera cam = new PerspectiveCamera(new Point(x:0, y:4, -10), Point.ORIGIN, new Vec3(x:0, y:5, z:0), distance:3, width:4, height:3);
        final Accelerator accel = new SimpleAccelerator();

        {
```

Overview - src Folder



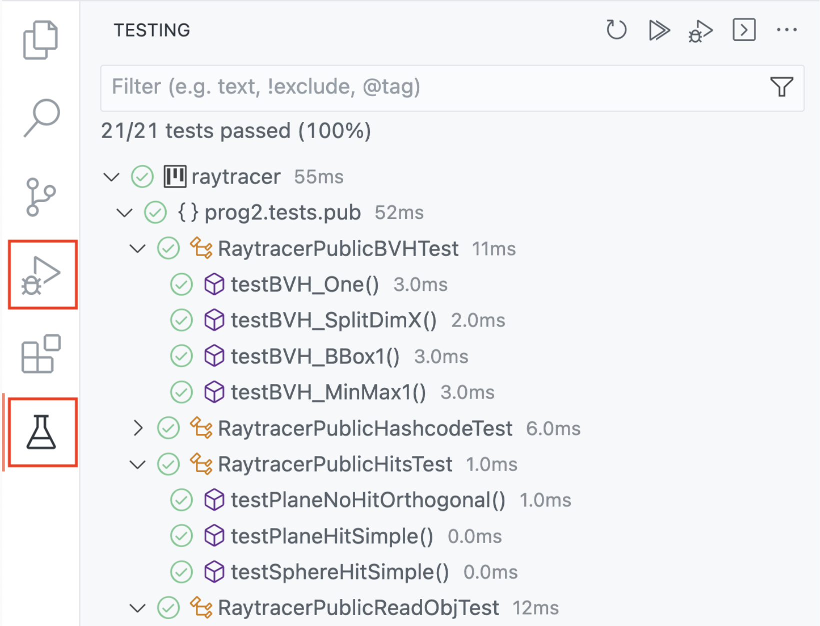
```
src / raytracer
  core
    def
      Accelerator.java
      BVH.java
      BVHBase.java
      LazyHitTest.java
      Main.java
      PointLightSource.java
      SimpleAccelerator.java
      StandardObj.java
      StandardScene.java
    Camera.java
    Hit.java
    LightSource.java
    Obj.java
    OBJReader.java
    PerspectiveCamera.java
    Renderer.java
    Scene.java
    Shader.java
    Trace.java
  geom
    BBox.java
    BBoxedPrimitive.java
    GeomFactory.java
    Plane.java
    Primitive.java
    Sphere.java
    Triangle.java
    Util.java
  math
  shade
    CheckerBoard.java
    Phong.java
    ShaderFactory.java
    SingleColor.java
```

Compiling and Running



```
(-10, y:10, -10), Color.WHITE);

), y:4, -10), Point.ORIGIN, new Vec3(x:0, y:5, z:0),
```

Testing and Debugging



TESTING

Filter (e.g. text, !exclude, @tag)

21/21 tests passed (100%)

```
✓ raytracer 55ms
  ✓ { } prog2.tests.pub 52ms
    ✓ RaytracerPublicBVHTest 11ms
      ✓ testBVH_One() 3.0ms
      ✓ testBVH_SplitDimX() 2.0ms
      ✓ testBVH_BBox1() 3.0ms
      ✓ testBVH_MinMax1() 3.0ms
    ✓ RaytracerPublicHashcodeTest 6.0ms
    ✓ RaytracerPublicHitsTest 1.0ms
      ✓ testPlaneNoHitOrthogonal() 1.0ms
      ✓ testPlaneHitSimple() 0.0ms
      ✓ testSphereHitSimple() 0.0ms
    ✓ RaytracerPublicReadObjTest 12ms
```

Unclear what a method is supposed to do?

Check the comments!

Questions?

Linear Algebra

Points and Vectors

- Points
    - denoted by lowercase characters (a)
    - three numbers describe location in 3D coordinate system
- Vectors
    - denoted by lowercase characters with arrow ($\overrightarrow{a}$)
    - three numbers describe displacement in 3D coordinate system

Vectors

Element-wise sum

$$\begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix} + \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix} = \begin{pmatrix} a_1 + b_1 \\ a_2 + b_2 \\ a_3 + b_3 \end{pmatrix}$$

gives a combined displacement

Vectors

Element-wise difference

$$\begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix} - \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix} = \begin{pmatrix} a_1 - b_1 \\ a_2 - b_2 \\ a_3 - b_3 \end{pmatrix}$$

gives the direct path from b to a

Vectors

Element-wise multiplication with scalar

$$k \begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix} = \begin{pmatrix} ka_1 \\ ka_2 \\ ka_3 \end{pmatrix}$$

Compresses ($|k| < 1$) or stretches ($|k| > 1$) $\overrightarrow{a}$

Vectors

Length of a vector calculated according to Pythagoras' theorem

$$|\overrightarrow{a}| = \sqrt{a_1^2 + a_2^2 + a_3^2} = \sqrt{\overrightarrow{a}^2}$$

Vector operations

Scalar Product

$$\vec{a} \cdot \vec{b} = |\vec{a}||\vec{b}|\cos\alpha = \begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix} \cdot \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix} = a_1 b_1 + a_2 b_2 + a_3 b_3$$

measures the enclosed angle
Special cases:

- 1: parallel, point in the same direction
- -1: parallel, point in opposite directions
- 0: perpendicular

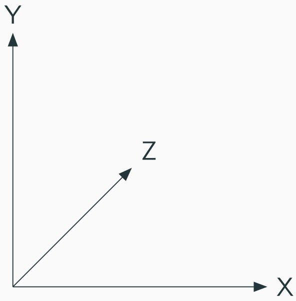
Vector operations

Cross Product

$$\overrightarrow{a} \times \overrightarrow{b} = \begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix} \times \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix} = \begin{pmatrix} a_2 b_3 - a_3 b_2 \\ a_3 b_1 - a_1 b_3 \\ a_1 b_2 - a_2 b_1 \end{pmatrix}$$

calculates a third vector that is perpendicular to the first two

Coordinate System



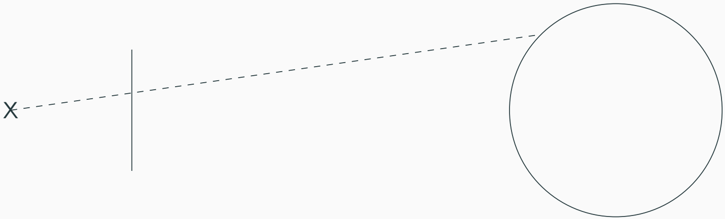
Left hand rule

- Thumb points in x-direction
- Index finger points in y-direction
- Middle finger points in z-direction

Questions?

Modeling

Camera



- Rays of light are looked at in reverse
- Observer modelled as camera with projection plane
- Viewing rays are sent out
- Hit points are determined
- Colour of hit points is calculated

Objects

- Rays (pre-implemented)
- Planes
- Triangles (pre-implemented)
- Spheres

Rays

- Uniquely described by a point p_s and a vector $\overrightarrow{v}_s$
- p_s is the starting point of the rays
- $\overrightarrow{v}_s$ is the direction of the rays (and a unit vector)
- λ is the distance from the starting point
- $x = p_s + \lambda \overrightarrow{v}_s$, $\lambda \geq 0$

Plane

- Uniquely described by three points (p_e, q_e, r_e) that are not on a straight line
- Hesse normal form $x\overrightarrow{n}_e - d = 0$ is better for intersecting
- The Hesse normal form is obtained from three points as follows:
  - $\overrightarrow{n'}_e = (q_e - p_e) \times (r_e - p_e)$
  - $\overrightarrow{n}_e = \overrightarrow{n'}_e \frac{1}{|\overrightarrow{n'}_e|}$
  - $d = p_e \overrightarrow{n}_e = |p_e| \frac{d}{|p_e|}$

Plane

- Intersection with ray is denoted by λ
- $\lambda = \frac{d - p_s \overrightarrow{n}_e}{\overrightarrow{v}_s \overrightarrow{n}_e}$, $\overrightarrow{v}_s \overrightarrow{n}_e \neq 0 \wedge \lambda \geq 0$
- If the denominator is 0: ray and plane are parallel
- Else λ is the distance from the starting point of the ray to where the hit is
- If $\lambda < 0$ the hit point is before the starting point of the ray, and thus invalid

Sphere

- Uniquely determined by a center point c_k and a radius r_k
- $|x - c_k| = r_k$
- alternatively, since $r \geq 0$: $(x - c_k)^2 = r_k^2$
- Intersection with ray is denoted by λ
- $b = 2\overrightarrow{v}_s(p_s - c_k)$, $c = (p_s - c_k)^2 - r_k^2$
- $\lambda_{1,2} = \frac{-b \pm \sqrt{b^2 - 4c}}{2}$
- If the value under the root is non-negative, the ray and the sphere intersect.
- The smaller λ is the first hit, the bigger hits the back of the sphere
- If $\lambda < 0$ the hit point is before the starting point of the ray, and thus invalid

Bounding Box

- Cuboid that encloses objects
- Infinite for plane
- Aligned with coordinate system axes
- Used to accelerate computation

Implementation

- use pre-implemented methods `geom.Util.computePlaneUV()` / `geom.Util.computeSphereUV()` to implement `getUV()`
- In hitTest() arguments t_{min} and t_{max} are passed, which describe the range in which a hit is "good"
- Look at the `Triangle` class to help you implement `Plane` and `Sphere`

Questions?

Light Sources

Light Sources

- Illuminate the scene
- Point light sources emit light rays uniformly from a single point in all directions
- Ambient light sources uniformly illuminate everything
- Shaders give objects their appearance
    - Simple Colour (pre-implemented)
    - Checkerboard
    - Phong

Checkerboard

- Alternates two other shaders on a grid
- Calculate which shader to use with coordinates u, v and scaling factor s
- $x = \lfloor u/s \rfloor + \lfloor v/s \rfloor$
- x is even: use the first shader
- x is odd: use the second shader

Phong

Consists of three parts: $I_{Phong} = I_{ambient} + I_{diffuse} + I_{specular}$

- $I_{ambient}$: Fixed colour added to everything to account for ambient light
- $I_{diffuse}$: Simulates scattering from a rough surface
- $I_{specular}$: Simulates specular reflection of light

Phong

$$I_{diffuse} = \sum_{l \in L} (c_l * c_{sub}) k_{diffuse} max(0, \overrightarrow{n} \overrightarrow{v})$$

- c_l: colour of light source
- c_{sub}: colour of the underlying shader
- $k_{diffuse}$: material constant
- $\overrightarrow{n}$: normal vector of the surface
- $\overrightarrow{v}$: vector to the light source

Phong

$I_{specular} = \sum_{l \in L} c_l k_{specular} max(0, \overrightarrow{r}\overrightarrow{v})^n$

- c_l: colour of light sources
- $k_{specular}$: material constant
- $\overrightarrow{r}$: reflection vector (obtained by mirroring on the normal)
- $\overrightarrow{v}$: vector to the light source
- n: gloss factor

Questions?

Acceleration Structure

- Help speed up the Raytracer
- If the bounding box of an object is not hit, then the object is not hit
- Objects are hierarchically sorted into bounding boxes to form a Bounding Volume Hierachry (BVH)
- Only a logarithmic number of checks necessary

Acceleration Structure

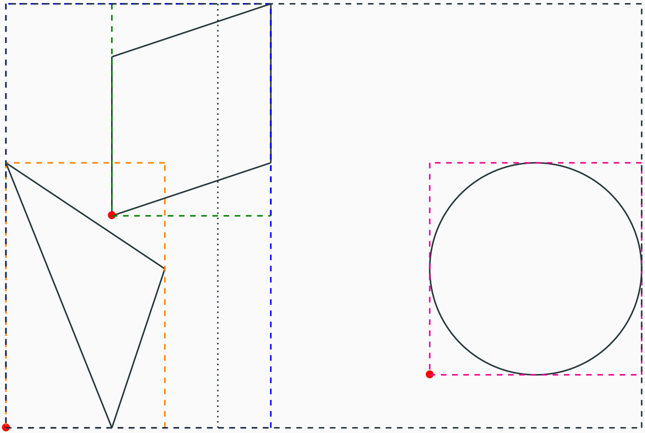


Figure 1: Minimal axis-aligned bounding boxes for two-dimensional shapes

Acceleration Structure

- `calculateMaxOfMinPoints`: compute the component-wise maximum of the minimum points for all objects
- `calculateSplitDimension`: determines in which dimension the box should be split
- `distributeObjects`: split the box and its objects
- `add`: add an object to the bounding box, adjust the corresponding bounding box
- The algorithm terminates as soon as 4 or less objects are in a box

Questions?

OBJ reader

- Describe polygon models
- Lines start with `v` or `f` to denote vertices and faces, respectively
- Vertices: three floats specifying the coordinates of the point
- Faces: integers specifying the vertices the face is made up of
- In our case, all faces are triangles

```
# comment
v 0.0 0.0 0.0
v 1.0 0.0 0.0
v 0.0 1.0 0.0
v 1.0 1.0 0.0
f 1 2 3
f 3 2 4
```

OBJ reader

- Lines not starting with v or f can be ignored
- Use `java.util.Scanner` to read the file
- call `useLocale()` with `Locale.ENGLISH` to make sure the points are recognized as decimal separators

Questions?

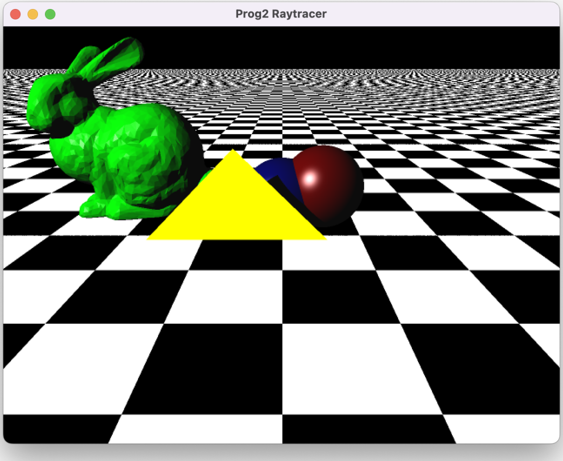
Running the Project

Running the Project

```
final boolean implementedPlane = true;
final boolean implementedCheckerBoard = true;
final boolean implementedSphere = true;
final boolean implementedPhong = true;
final boolean implementedOBJReader = true;
final boolean implementedBVH = true;
```

Set the flags to true to make things appear in the reference image

Reference image



Questions?