Objektorientierte Programmierung mit einem Raytracer

Projekt in Eclipse importieren

Da wir jetzt eine externe Funktionalität benutzen wollen, müssen wir diese Funktionen zuerst in Eclipse importieren:

1. Zuerst klicke mit der rechten Maustaste in die Projektübersicht und wähle die Funktion `Import...`
2. Wähle dann unter `General`: `Existing Projects into Workspace` und klicke auf `Next >`
3. Wähle aus dem Tauschlaufwerk im Projektverzeichnis für den Informatikkurs den Ordner `Raytracing` aus.
4. Unbedingt den Haken bei `Copy projects into workspace` setzen!
5. Mit einem Klick auf `Finish` wird das Projekt importiert

Eigenes Projekt anlegen

In unserem eigenen Projekt wollen wir die Funktionen des `Raytracing`-Projekts nutzen und müssen diese deshalb angeben wenn wir unser Projekt erstellen:

1. wähle wie bisher im Menü `File`→ `New`→ `Java Project`
2. gib den Namen `OOP` ein, wir werden die kommenden Wochen an diesem Projekt arbeiten und dieses weiterentwickeln
3. klicke **nicht** auf `Finish` sondern auf `Next >`
4. wähle im darauffolgenden Dialog `Projects` und füge das `Raytracing`-Projekt hinzu.
5. Mit einem Klick auf `Finish` wird das Projekt importiert

Damit können wir in unserem Projekt `OOP` die Funktionen des `Raytracing`-Paketes benutzen.
Für die bessere Strukturierung lege in dem Projekt ein Paket `start` an und darin eine Klasse `Start` (diese wieder mit der `main`-Methode)

Raytracer benutzen

Um den Raytracer benutzen zu können müssen wir die Pakete importieren mit `import raytracing.*;`
Anschließend legen wir in der `main`-Methode den Raytracer an mit

```
public static void main(String[] args) {
    Tracer tr = new Tracer();
}
```

Listing 1: Anlegen des Raytracers

Wenn wir so das Programm ausführen, so öffnet sich nur ein leeres, schwarzes Fenster.
Mit der Methode `tr.setPixel( x , y , r , g , b );` können wir einen einzelnen Pixel an der Koordinate $(x \mid y)$ auf einen RGB-Farbwert (r,g,b) setzen.
Hierbei ist zu beachten, dass die x-Koordinate wie gewohnt von ganz links $(x = 0)$ bis ganz rechts hochgezählt wird, die y-Koordinate jedoch von oben $(y = 0)$ nach unten hochgezählt wird! Die Fensterbreite bzw. -höhe bekommen wir mit Methode `tr.getWidth()` bzw. `tr.getHeight()`.
Die RGB-Farbwerte liegen jeweils zwischen 0 (dunkel) und 1 (volle Farbe).

1. Aufgabe

Lege das Projekt an und zeichne manuell den Anfangsbuchstaben von deinem Namen in das Fenster, indem du die einzelnen Pixel einfärbst.

2. Aufgabe

a) Lasse (mithilfe einer `for`-Schleife) eine Zeile des Fensters einfärben

b) Lasse (mithilfe einer `for`-Schleife) eine Spalte des Fensters einfärben

c) Kombiniere diese beiden Schleifen um das ganze Fenster einzufärben

d) Probiere auch unterschiedliche Farben selbst aus um dich mit dem RGB-Farbschema vertraut zu machen.

e) *Zusatzaufgabe:* Färbe das Fenster so ein, dass der Pixel in der linken oberen Ecke schwarz ist, und der Rotwert nach rechts zunimmt bis er auf der rechten Seite dann bei $r = 1$ ist. Nach unten soll der Grünwert gleichermaßen zunehmen.

Objekte sichtbar machen

In dem virtuellen Raum im Fenster (diesen nennt man auch *Szene*) sind auch einige Objekte versteckt. Du kannst die Methode `tr.trifft( x , y )` benutzen um herauszufinden, ob ein Lichstrahl, der vom Auge ausgeht und durch den Pixel $(x \mid y)$ geht, ein Objekt in der Szene trifft. Die Methode liefert als Ergebnis also einen `boolean`-Wert zurück den wir mit einer `if`-Bedingung abfragen können.

3. Aufgabe

Benutze die `for`-Schleifen von oben, um jeden Pixel des Fensters zu durchlaufen. Teste damit jeden Pixel auf einen Treffer mit einem Objekt und setze den Pixel bei einem Treffer auf eine Farbe.

4. Aufgabe

Neben der Methode `tr.trifft( x , y )` kannst du auch die Methoden `tr.rot( x , y )`, `tr.gruen( x , y )` und `tr.blau( x , y )` benutzen. Diese liefern – sofern ein Objekt getroffen wird – als Ergebnis jeweils einen `double`-Wert mit der jeweiligen RGB-Farbkomponente.
Benutze diese, um die Objekte der Szene in der passenden Farbe anzuzeigen.