# Aufgabe Schnittpunktest optimieren

Der in `triangle.tcc` implementierte Schnittpunkttest, soll durch eine optimierte Variante ergänzt werden. Es sollen folgende Berechnungen verzögert und eingespart werden:

- Falls der Parameter `t` nicht kleiner als der bisherige Schnittpunktparameter `minimum_t` ist, soll frühzeitig abgebrochen werden.
- Die u-v-Parameter sollen erst am Ende der Funktion berechnet werden inklusive der dazu notwendigen Zwischenergebnisse.
- Statt drei mal die Quadratwurzel über den Aufruf der `lenght()`-Methode zu berechnen, soll sie nur zwei mal berechnet werden. Dazu muss das Quadrat der Länge mit `square_of_length()` verwendet werden und die Berechnung der u-v-Parameter geeignet umgeformt werden.

## Aufgabe 1

Implementieren Sie die optimierte Variante

```
bool intersects(Vector<T,3> origin, Vector<T,3> direction,
                FLOAT &t, FLOAT &u, FLOAT &v, FLOAT minimum_t)
```

Sie befindet sich nach der `#else`-Präprozessor-Direktive.

## Aufgabe 2

Übersetzen Sie das Programm für die Zeitmessung auf den Rechnern in E203 mit

```
g++ -Wall -pedantic -march=native -mfpmath=sse -mavx -O3 raytracer.cc statistics.cc
```

Messen Sie die Ausführungszeiten 10 mal mit dem übersetzen Programm und bilden sie den Durchschnittswert.

## Aufgabe 3

Erstellen Sie einen Bericht zu dieser Aufgabe mit dem Quelltext Ihrer Funktionen, den zugehörigem Teilen des Assemblercode und einer tabellarischen Gegenüberstellung der Ausführungszeiten.

Interpretieren Sie die Resultate. Wenn die Zeiten nahezu gleich sind: Warum sind sie gleich? Wenn eine Version deutlich schneller ist: Warum sind ist sie schneller als eine andere?