Grundlagen der C++-Programmierung

Assignment due Wednesday June 13 (23:59)

Spaceships – Tests

Pflichtaufgabe

Ziel dieser Aufgabe ist es, mit und anhand von Tests zu implementieren.

Ihr wollte ein Spiel (oder eine ernsthaftere Anwendung) entwerfen, wo Raumschiffe sich selbst und Planeten zerstören können. Eine grobe Skizze dazu gibt es schon, die findet Ihr in den Kommentaren in der Datei `main.cpp`.

Die Vorgaben sind absichtlich vage, denn Ihr sollt selbst frei über Schnittstellen/-Signaturen und Implementierung nachdenken. Lediglich die gewünschte Funktionalität ist grob vorgegeben. Versucht, aus dieser Vorgabe Tests zu formulieren, mit deren Hilfe Ihr die Implementierung treibt und dabei Fehler möglichst ausschließt.

Die Tests dienen zum frühen Erkennen von Fehlern bzw. können verwendet werden, wann immer etwas erweitert wird: danach muss die "alte Funktionalität" immer noch gegeben sein. Ihr könnt auf diese Weise mit einer "leeren Hülle" anfangen und diese schrittweise verfeinern.

Dazu müsst Ihr folgende Schritte (am besten in dieser Reihenfolge) umsetzen:

1. Definiert ein einfaches **Testframework** durch eine Funktion `TEST()` (s.u.).
2. **Definiert** die in `main.cpp` geforderten **Tests** mit Hilfe von `TEST()` um.
3. **Implementiert** die geforderte **Funktionalität**, so dass alle Tests erfolgreich sind.

Die Funktion `TEST(expr,description)` soll mindestens zwei Argumente nehmen: Sie testet ob der Ausdruck `expr` wahr ist. Wenn ja wird (in einer Zeile) `PASSED $desc"` ansonsten `FAILED $$desc"` nach `cerr` **ausgegeben**. Dabei steht `$$desc` für eine kurze Beschreibung `description` z.B. so wie in `main.cpp` gegeben.

Das ganze soll so ähnlich aussehen wie die Tests, die wir verwenden und die Ihr aus dem Backend kennt. Deshalb soll am Ende noch in einer Zeile ein **Gesamtreport** ausgegeben werden, der die Anzahl erfolgloser Tests und die Gesamtanzahl ausgibt und entsprechend `PASSED"` oder `FAILED"`.

Wenn alle Tests erfolgreich sind, soll Eure `main()`-Funktion den Wert `0` zurückgeben, ansonsten einen anderen Wert. Ihr könnt dazu auch exit verwenden.

Zum Nachdenken: Wie könnte man den Gesamtreport automatisch zum Programmende ausgeben, ohne nochmal eine Funktion/Methode explizit aufrufen zu müssen?

Optional (nochmal zum Nachdenken)

Es bietet sich an, `TEST()` als ein **Macro** zu implementieren, das eine Funktion aufruft. Auf diese Art und Weise könnt Ihr folgende Zusatzinformationen ausgeben, falls ein Test fehl schlägt:

- Dateiname und Zeile im Quellcode, wo es schief gegangen ist;

- ggf. dazu die Funktion (Vorsicht: Kein Standard! Muss für alle Compiler funktioneren!);
- Nochmal den Ausdruck `expr`, der `false` liefert, als Text.

Um das umzusetzen, müsst Ihr Euch mit Macros und dem Präprozessor auseinandersetzen!