Leibniz Universität Hannover
FG Graphische Datenverarbeitung
Prof. Dr. F.-E. Wolter
Dipl.-Math. Andreas Tarnowsky

9th February 2016
Labor Grafikprogrammierung
Übungsblatt 0

Labor Grafikprogrammierung

Übungsblatt 0

Über die Vergabe der Laborplätze

Da nur eine begrenzte Anzahl von Arbeitsplätzen zur Verfügung steht, werden in diesem Jahr die Laborplätze anhand der Abgabe dieses Übungszettels vergeben. Neben Korrektheit der Implementation wird auch die Kreativität der Abgabe berücksichtigt - die weiter unten angegebenen Bonusaufgaben sollen nur als Anreiz für eigene Ideen dienen. Nach Abgabeschluss werden alle Einsender darüber informiert, ob sie für die Veranstaltung zugelassen wurden. Eventuell übrige Laborplätze werden danach zur öffentlichen Anmeldung freigegeben. Übersteigt die Zahl der Anmeldungen die der verfügbaren (Rest-)Plätze so entscheidet das Los.

Das GDV-Framework

Für die Bearbeitung der Aufgaben wird ein Framework bereitgestellt, welches das Programmieren in der Sprache *C++* möglichst einfach gestalten soll. Um den Einstieg zu erleichtern, befindet sich im Ordner *examples* eine kommentierte Beispielklasse, die die grundlegende Funktionalität des Frameworks erklärt. Die Verwendung des Frameworks ist *verpflichtend*. Das GDV-Framework können Sie über folgenden Link herunterladen:

`http://data.welfenlab.de/lectures/grafikprogrammierung/GDV-Framework.zip`

Als Programmierumgebung wird *QtCreator* empfohlen, welche unter

`http://www.qt.io/download-open-source/`

bezogen werden kann.

Abgabe

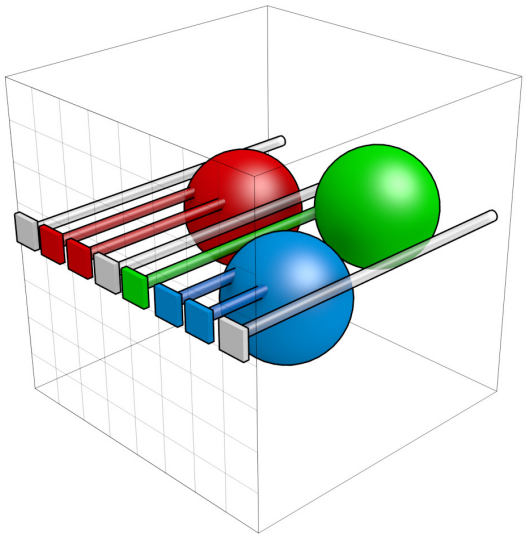
Bitte senden Sie Ihre fertige Abgabe (nur die von Ihnen erstellten Klassen, *nicht* das gesamte Framework) an

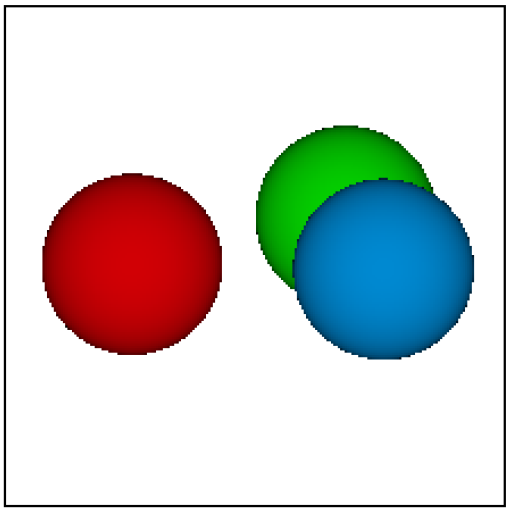
`atarnows@welfenlab.de`

Abgabeschluss ist der 31.03.2016

Aufgabe 1

Implementieren Sie einen einfachen *Raycaster* (eine vereinfachte Variante eines Raytracers), der in der Lage ist Kugeln verschiedener Größe und Farbe zu rendern. Positionieren Sie eine variable Anzahl solcher Kugeln zufällig im Raum. (Sie können der GUI einen Knopf hinzufügen, der die Positionen der Kugeln zufällig neu setzt.)
Das folgende Bild (links) zeigt die prinzipielle Vorgehensweise:





Für jedes Pixel des Bildschirms wird ein Strahl erstellt der orthogonal zur Bildschirmebene liegt. Schneidet dieser Strahl eine der Kugeln, so wird die Farbe derjenigen Kugel übernommen, welche den Strahl *zuerst* schneidet. Um eine einfache Beleuchtung der Kugeln zu erreichen, wird der *Richtungsvektor* mit der *Normalen* der Kugel am jeweiligen Schnittpunkt mit dem Skalarprodukt verrechnet. Der resultierende Wert wird nun verwendet, um die Helligkeit des Pixels zu bestimmen – die Helligkeit kann multiplikativ mit dem Farbwert verrechnet werden.
Das Resultat dieses einfachen Raycasters könnte beispielsweise wie im rechten Bild aussehen.

Aufgabe 2

Erweitern Sie den Raycaster aus Aufgabe 1 nach eigenem Ermessen.
Die folgenden Techniken wären beispielsweise denkbar:

- (Zufällig positionierte, farbige, etc.) Punktlichtquellen
- Darstellung weiterer geometrischer Objekte (Boxen, Kegel, Zylinder, Torus, etc.)
- Supersampling
  Siehe `http://paulbourke.net/miscellaneous/aliasing/`
- Transparenz
- Schattenwurf, Reflektionen, Lichtbrechung, ...