Media Informatics /
Graphics & Geometric Computing

Prof. Dr. Marcel Campen
Axel Schaffland, M.Sc.

UNIVERSITÄT OSNABRÜCK

Computergrafik (SS 2018)

Übung 3

fällig Montag **30. April**, 14:20

- Geben Sie bei jeder Abgabe alle Ihre Matrikelnummern auf jedem Blatt an.
- Verspätet eingereichte Abgaben können nicht gewertet werden.

Aufgabe 1 (Theorie: Frustrums, Linien, Dreiecke)

(a) (3P) Frustrum-Matrix:

1. Geben Sie die Frustrum-Matrix für einen horizontalen Öffnungswinkel (*fovx*) von $90°$ und einen vertikalen Öffnungswinkel (*fovy*) von $90°$ mit near-Plane $n = 1$ und far-Plane $f = 4$ an.
2. Geben Sie die Koordinaten von zwei der acht Eckpunkte dieses Frustrums aus (a) (des Pyramidenstumpfes, der den sichtbaren Bereich vor der Kamera begrenzt) an, davon ein Punkt in der near-Plane, ein Punkt in der far-Plane.
3. Wenden Sie die Frustrum-Matrix aus (a) auf diese beiden Punkte aus (b) an und konktrollieren Sie, ob diese (ggf. nach Dehomogenisierung), wie erwartet, Ecken des Würfels $[-1, 1] \times [-1, 1] \times [-1, 1]$ bilden.

(b) (1P) Der Punkt $(3, 3)$ liegt auf der Gerade durch $(1, 2)$ und $(7, 5)$. Stellen Sie ihn als Linearkombination dieser beiden Punkte dar.

(c) (2P) Berechnen Sie die baryzentrischen Koordinaten des Punktes $(0, 0)$ bezüglich des Dreiecks mit den Ecken (in Reihenfolge gegen den Uhrzeigersinn) $(2, 1)$, $(3, 3)$, $(1, 2)$.

(d) (2P) Markieren Sie in einem Pixelraster die Pixel, die nach den Regeln des Bresenham-Algorithmus gefärbt werden, wenn eine Linie vom Punkt $(4, 10)$ zum Punkt $(0, 0)$ gezeichnet wird.

Aufgabe 2 (Praxis: Linien und Dreiecke)

Schauen Sie sich den Code für Übungsblatt 3 an und öffnen Sie ihn in Ihrem Browser.

Erledigen Sie die folgenden Aufgaben indem Sie die mit `BEGIN` und `END` markierten Leerstellen im Code ausfüllen. Verändern Sie den übrigen vorhandenen Code nicht.

Hinweis: In JavaScript können Sie sich (im Rahmen der Leerstellen) weitere Hilfsfunktionen innerhalb der vorgegebenen Funktionen definieren, um Ihren Code ggf. eleganter zu gestalten.

Hinweis: Sie können die Größe des Canvas mit der Konstanten `size` anpassen, um ggf. das Rendering zu beschleunigen.

(a) 1. (1P) Vervollständigen Sie die Funktion `createLines`: Erstellen Sie mehrere Liniensegmente, so dass diese einen beliebigen dreidimensionalen Körper (z.B. einen Würfel) formen. Zur Verwaltung der Linien wird Ihnen die Klasse `Line3D` zu Verfügung gestellt. Die Linien sollen im Array `linearray` gespeichert werden. Linien werden spezifiziert durch einen Start- und einen Endpunkt. Für Punkte wird die bekannte `Point3D`-Klasse verwendet. Eine Linie erstellen Sie durch `new Line3D(point1, point2)`. Die Koordinaten der Punkte sollten zwischen -200 und +200 liegen. Anschließend sollten Ihnen noch nicht die Linien, jedoch bereits ihre Endpunkte angezeigt werden (aufgrund des Codes im Bereich `TEST CODE`). Per Maus lässt sich die Ansicht rotieren ("Virtual Trackball").

2. (5P) Vervollständigen Sie die Funktion `drawLine` (welche vom vorhandenen Code für jede Linie (nach Transformation, Projektion, Viewport-Transform) aufgerufen wird): Zeichnen Sie eine Linie, indem Sie den Bresenham-Algorithmus (oder eine vereinfachte (etwas weniger effiziente) Variante, wie in der Vorlesung vorgestellt) implementieren. Achten Sie darauf zunächst die Richtung und Steigung der Linie zu bestimmen und entsprechend darauf zu reagieren. Start- und Endpunkt der Linie stehen in den Variablen `startpoint` und `endpoint` zu Verfügung. Das Pixel (x, y) färben Sie schwarz, indem Sie `drawPoint(x,y)` aufrufen. Wenn die Checkbox "Zeichne Linien" aktiv ist, sollten Sie nun die Linien sehen.

(b) 1. (2P) Vervollständigen Sie die Funktion `createTriangles`: Erstellen Sie mehrere Dreiecke, so dass diese einen beliebigen dreidimensionalen Körper (z.B. einen Würfel) formen. Die Dreiecke sollen in dem Array `trianglearray` gespeichert werden. Ein Dreieck erstellen Sie durch `new Triangle3D(point1, point2, point3, color)`. Jedes Dreieck besteht aus den drei Eckpunkten vom Typ `Point3D` sowie einem Array `color` mit drei Einträgen [r,g,b] für die Farbinformationen. Auf diese Weise können Sie jedem Dreieck eine Farbe zuweisen, um diese später in der Darstellung besser auseinanderhalten zu können. Der Wertebereich für die Punkte sollte wieder zwischen -200 und +200 liegen, der für Farbwerte im Bereich zwischen 0 und 255.

2. (4P) Vervollständigen Sie die Funktion `drawTriangle` (welche vom vorhandenen Code für jedes Dreieck (nach Transformation, Projektion, Viewport-Transform) aufgerufen wird): Zeichnen Sie ein Dreieck, indem Sie die Pixel, deren Mittelpunkt innerhalb des Dreiecks liegt mit der Farbe des Dreiecks einfärben. Nutzen Sie dazu `drawPoint(x, y, color)`. Testen Sie nicht naiv für jedes Pixel des Canvas, ob es innerhalb des Dreiecks liegt, sondern beschränken Sie den Test auf eine "BoundingBox" des Dreiecks.

   Wenn die Checkbox "Zeichne Dreiecke" aktiv ist, sollten Sie nun die Dreiecke sehen.

   Hinweis: Werden mehrere Dreiecke auf die gleiche Stelle projiziert, sehen Sie dort ggf. nicht das vorderste, sondern das zuletzt verarbeitete. Mit der Checkbox "Sortiere Dreiecke" werden die Dreiecke automatisch von hinten nach vorne sortiert, bevor Ihre Funktion `drawTriangle` aufgerufen wird. Dies schafft (teilweise) Abhilfe – vollständig korrekte Lösungen für dieses Problem schauen wir uns später in der Vorlesung an.

Aufgabe 3 (Bonus: Antialiasing von Linien)

Modifizieren Sie den Code so, dass die Linien mit Antialiasing gezeichnet werden.

Für die Bonusaufgabe darf der Code auch jenseits der markierten Bereiche angepasst werden.