Übung zu Computergraphik I

– Übungsblatt 11 –

Lehrstuhl für Computergraphik und Multimediasysteme

Andreas Görlitz, Hendrik Hochstetter, John Rickard, Rene Winchenbach

Abgabe: Für Studenten mit 5 LP verpflichtend bis spätestens Dienstag 19. Januar 2016, 10 Uhr

Besprechung: Dienstag 26. Januar 2016 und Mittwoch 27. Januar 2016

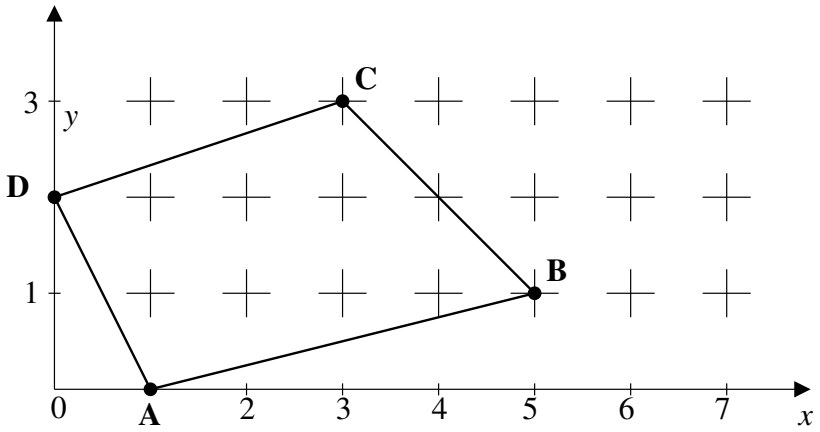
Hinweise: Bearbeitungen bitte mit Name, Matrikelnummer und Übungsgruppe beschriften und zusammengeheftet in den Briefkasten vor Büro H-A 7115/1 werfen. Programmieraufgaben bitte per Mail mit Name und Matrikelnummer an Ihren jeweiligen Tutor senden. Geben Sie bei dabei nur Ihre modifizierte Quelltextdatei als Anhang ab.

Aufgabe 1 Polygonrasterisierung - Scanline-Algorithmus (1 Punkt)

Gegeben seien die folgenden vier Punkte eines Polygons in Raster-Koordinaten:

$$\mathbf{A} = \begin{pmatrix}1\\0\end{pmatrix}, \mathbf{B} = \begin{pmatrix}5\\1\end{pmatrix}, \mathbf{C} = \begin{pmatrix}3\\3\end{pmatrix}, \mathbf{D} = \begin{pmatrix}0\\2\end{pmatrix}$$

Im Folgenden soll das Polygon mit Hilfe des Scanline-Algorithmus rasterisiert werden.



1.1 Berechnen Sie die Anfangs- und Endpunkte aller Spans, indem Sie den Scanline-Algorithmus durchführen.

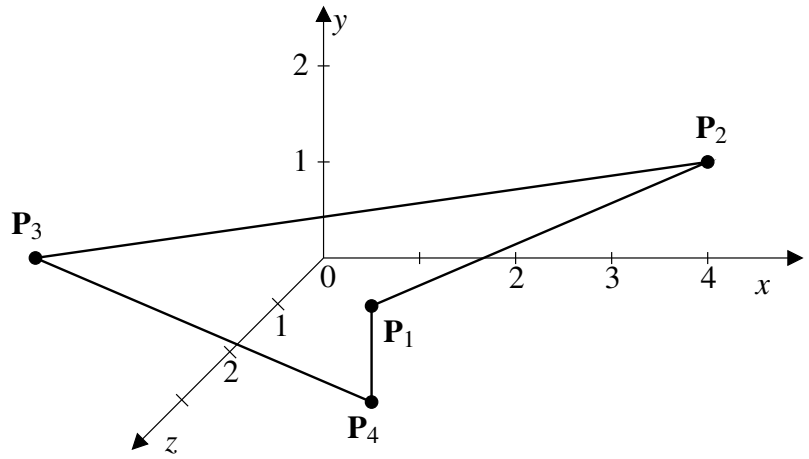
1.2 Zeichnen Sie die Punkte der Polygonrasterisierung in die Abbildung ein.

Aufgabe 2 Point-in-Polygon-Test (1 Punkt)

Im Raycasting-Verfahren wird für jeden Bildpunkt ein Strahl mit den Objekten der Szene geschnitten. Im Folgenden soll ein Point-in-Polygon-Test durchgeführt werden, um zu prüfen, ob ein Strahl ein Polygon mit vier Eckpunkten trifft.
Das Polygon sei definiert durch die folgenden vier Punkte, die in dieser Reihenfolge miteinander verbunden sind:

$$\mathbf{P}_1 = \begin{pmatrix}1\\0\\1\end{pmatrix}, \mathbf{P}_2 = \begin{pmatrix}4\\1\\0\end{pmatrix}, \mathbf{P}_3 = \begin{pmatrix}-2\\1\\2\end{pmatrix}, \mathbf{P}_4 = \begin{pmatrix}1\\-1\\1\end{pmatrix}$$

Wir betrachten einen Strahl g in z-Richtung, ausgehend vom Punkt $\mathbf{V} = (-2,0,0)^\mathsf{T}$.



2.1 Bestimmen Sie den Schnittpunkt $\mathbf{S}$ der Polygonebene mit dem Strahl g. Hinweis: Berechnen Sie hierfür den Normalenvektor $\hat{\mathbf{n}}$ der Polygonebene.

2.2 Projizieren Sie die Punkte $\mathbf{P}_1, \mathbf{P}_2, \mathbf{P}_3, \mathbf{P}_4$ und den Schnittpunkt $\mathbf{S}$ entlang der treibenden Achse des Normalenvektors $\hat{\mathbf{n}}$.

2.3 Führen Sie eine Verschiebung der Punkte durch, so dass der Schnittpunkt $\mathbf{S}$ auf den Ursprungspunkt verschoben wird.

2.4 Zeichnen Sie die transformierten Punkte in eine 2D-Skizze ein, und prüfen Sie graphisch und rechnerisch, ob $\mathbf{S}$ im Polygon enthalten ist. Hinweis: Betrachten Sie die Schnittpunkte entlang der x-Achse.

2.5 Für welche Polygonkante(n) reicht es aus, eine Trivial-Reject-Prüfung durchzuführen?

Aufgabe 3 Raycasting (Bonusaufgabe 1 Punkt)

In dieser Aufgabe sollen Sie einen Raycaster erweitern. Laden Sie hierzu zunächst das Programmgerüst `ueb11.zip` von der Übungswebseite herunter. Im Programm wird bereits das Objekt erzeugt und es stehen folgende Steuerungsmöglichkeiten zur Verfügung:

1. Mit gedrückter linker Maustaste rotieren Sie das Objekt um seinen Mittelpunkt.
2. Mit gedrückter rechter Maustaste zoomen Sie in die Szene.
3. Mit gedrückter mittlerer Maustaste verschieben Sie Ihren Fokus auf die Szene.
4. Mittels Leertaste kann die Ansicht auf die Initialeinstellung zurückgesetzt werden.

Abbildung 1 zeigt einen Screenshot des vollständigen implementierten Raycasters.

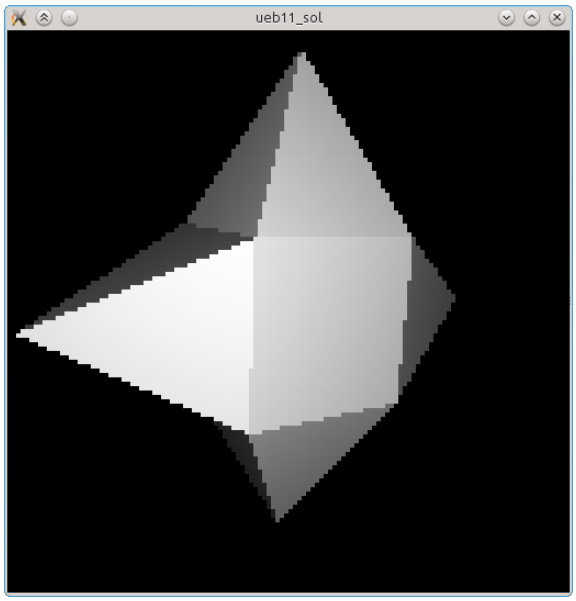


Abbildung 1: *Rendering eines sternenförmigen Objektes mittels Raycasting.*

Nur Strahlen, die ein Dreieck treffen, erzeugen einen sichtbaren Anteil im Bild. Hierzu wird für jeden Strahl die Methode `Raycaster::pointInPolygonTest()` aufgerufen. Implementieren Sie in dieser Funktion die Schritte für den Point-In-Polygon Test aus Aufgabe 2 auf Basis von Dreiecksprimitiven. Als Parameter bekommen Sie ein Dreieck `tri` vom Typ `Triangle` und den Schnittpunkt mit der Dreiecksebene `iPoint` vom Typ `vec3` übergeben. Geben Sie `true` im Falle dass der Schnittpunkt im Dreieck liegt oder andernfalls `false` zurück.

Sofern Sie alles richtig implementiert haben, sollte Ihr Ergebnis dem der Abbildung entsprechen. Im Urzustand bleibt das Fenster leider schwarz.