Prof. G. Zachmann
J. Boeckers
A. Mühlenbrock
N. M. Jadid

Universität Bremen
Fachbereich 3
Arbeitsgruppe CGVR
4. Dezember 2024

Wintersemester 2024/25

Übungen zu Computergrafik - Blatt 6

Abgabe am 01.12.2024, 23:59 Uhr

Aufgabe 1 (Rasterisierung einer Linie, *1,5+1 Punkte*)

a) In der Vorlesung wurde ein einfacher, inkrementeller Algorithmus zur Rasterisierung von Linien vorgestellt (Folien 7-9). Zeige anhand eines Rechenbeispiels, wie die ersten Linienparameter t_x^{next} und t_y^{next} sowie die Deltas Δt_x und Δt_y berechnet werden können. Gegeben sei dazu eine Linie von $P_0 = (1, 0)$ nach $P_1 = (4, 7)$. Hier kannst Du davon ausgehen, dass das Gitter bei P_0 die linke untere Ecke hat, und $g_x = g_y = 1$.

b) Wenn das Gitter nicht bei P_0 anfängt, welche Auswirkung hätte das auf die Anfangswerte von t_x^{next} und t_y^{next}, und ggf. Δt_x und Δt_y? Es genügt jeweils eine Erläuterung (ggf. mit Bild oder Zeichnung), es muss nicht ausgerechnet werden.

Aufgabe 2 (Interpolation, *1,5 Punkte*)

Wir entwickeln im Folgenden einen einfachen Algorithmus zur Vergrößerung von Bildern. Wir nehmen an, dass das Eingabebild ein Grauwert-Bild ist, gegeben als $I(x, y)$. Nehme nun an, dass die Pixelwerte von I auf einem Gitter gegeben sind, mit $x \in [0, w] \subset \mathbb{N}$, und $y \in [0, h] \subset \mathbb{N}$. Das Bild soll auf ein Bild $\hat{I}$ der Größe $[0, W] \times [0, H]$ vergrößert werden (siehe Abbildung 1). Der Grauwert von $\hat{I}(x, y)$ müsste also an der Stelle $I(\frac{w}{W}x, \frac{h}{H}y)$ ausgelesen werden. An dieser Stelle befindet sich aber kein Pixel, sondern an den vier Gitterpositionen darum herum. D.h. in dieser Aufgabe sind die Pixelwerte an den Gitterknoten gegeben, nicht den Mittelpunkten der Zellen!

Wie können die Koordinaten der umliegenden Eckpunkte berechnet werden, wenn diese ganzzahlig sind? Gebe an, wie man mittels Interpolation der Grauwerte der Eckpunkte den Grauwert $\hat{I}(x, y)$ des vergrößerten Bildes berechnen kann. Es genügt jeweils eine Erläuterung, ggf. mit Formel.

Aufgabe 3 (Continuous Collision Detection (CCD), *2,5+2,5+1 Punkte*)

a) In der Vorlesung haben wir das Voxmap-Pointshell Verfahren zur Continuous Collision Detection kennen gelernt. Der 1. Schritt besteht hierbei daraus, alle Voxel zu finden die vom Liniensegment getroffen werden, siehe die rechte Line in Abbildung 2. Dies ist in dem Framework „RasterFramework“ auf der CGVR Website teilimplementiert und soll vervollständigt werden. Dazu muss der mit „TODO 3a“ markierte Abschnitt in der Datei `main.cpp` bearbeitet werden. Orientiere dich an dem Algorithmus zum Rastern von Linien aus der Vorlesung. Für den Schnitttest muss zudem die Funktion `intersectPlane` vervollständigt werden. Die Farbe spielt vorerst keine Rolle.

   Tipp: Für einen einfachen Schnitttest zwischen Liniensegment und Ebene bietet es sich an, von der Linie in Parameterform und der Ebene in Normalenform auszugehen.

   Hinweis: Mittels dem Parameter `wf` in der Rendermethode des cube-meshes kann zu Testzwecken in den Wireframe-Modus geschaltet werden. Im GUI kann zudem zwischen 3 Ansichten gewechselt werden (front, top-down, side).

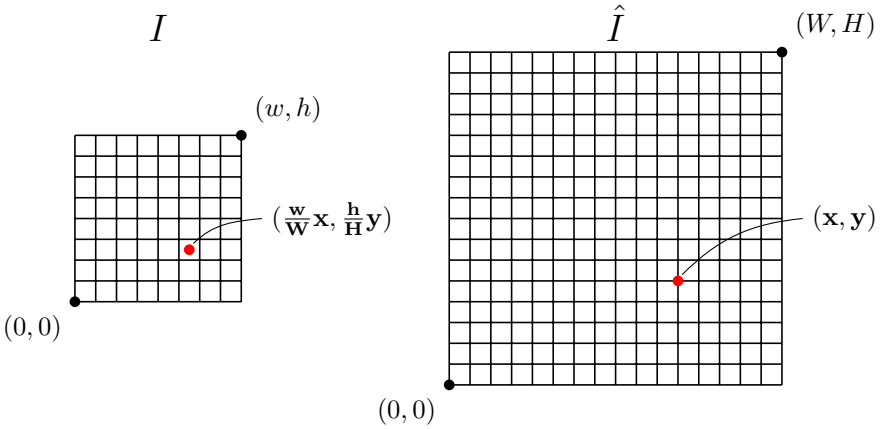

Abbildung 1: Situationsskizze zu Aufgabe 2

b) Implementiere nun das in der Vorlesung gezeigte Verfahren zur Interpolation von Attributen auf einer Linie. Nutze dazu wieder das zuvor genannte Framework „RasterFramework“. In der Datei `main.cpp` muss hierzu der mit „TODO 3b“ markierte Abschnitt bearbeitet werden. Die gerasterten Koordinaten einer Linie sind hier bereits vorgegeben. Deine Aufgabe ist es, die Farben der Linie zu interpolieren. Berechne dafür den Interpolationsfaktor inkrementell (d.h. nicht in jedem Schritt komplett neu!). Das Ergebnis sollte dem in Abbildung 2 (linke Linie) entsprechen.

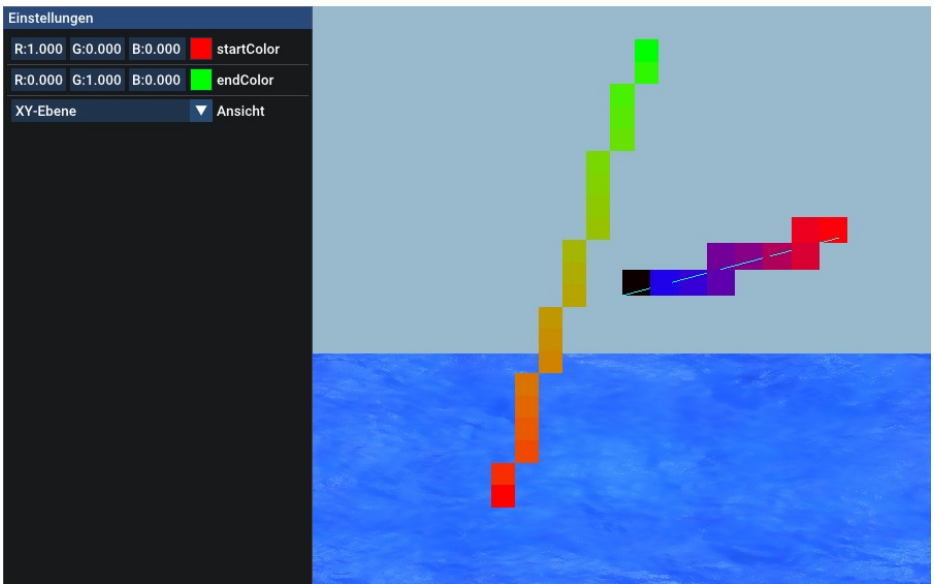

Abbildung 2: *Links*: Aufgabe 3 b): Die vorgegebene Linie mit interpolierten Farben. *Rechts*: Aufgabe 3 a): Die aus Würfeln gerasterte Linie, überlagert mit einer Debuglinie (Türkis).

c) Skizziere mindestens zwei unterschiedliche Fälle, in denen das in der Vorlesung genannte Ausschlusskriterium 2 zum Test auf Nulldurchgang in der Continuous Collision Detection fälschlicherweise ein negatives Ergebnis liefert (Folie 34).
Hinweis: Überlege, welche Form eine Oberfläche annehmen kann, je nach dem wie die Distanzen an den Eckpunkten verteilt sind (es gibt mehrere Möglichkeiten).