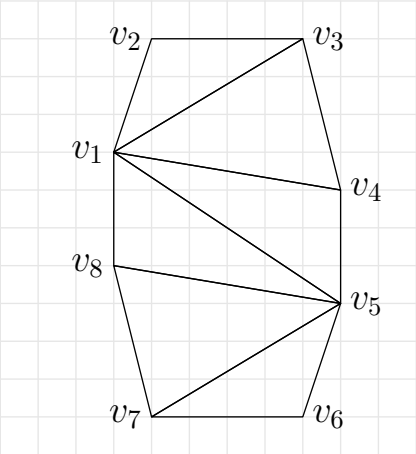
Farben, Geometrie, Rasterisierung

Kurz & Knapp

- Erläutern Sie knapp die Begriffe 'Retina' und 'Fovea'.
- Was ist der Unterschied zwischen Stäbchenzellen (Rods) und Zäpfchenzellen (Cones)?
- Was versteht man unter Tristimulus Farbtheorie?
- Wie stehen RGB und CMY in Bezug?
- Welcher Wertebereich und Binärrepräsentation wird beim Speichern von RGB Bildern (z.B. in Form von Pngs) verwendet? Gibt es Alternativen?
- Geben Sie zur CMY Farbe $(\frac{1}{3}, \frac{1}{4}, \frac{1}{5})$ die 'faulst mögliche' und die 'effizienteste' CMYK Farbe an.
- Beschreiben Sie knapp die Idee des SD-Raster Verfahrens.
- Beschreiben Sie knapp die Idee des Scanline Algorithmus.
- Erläutern Sie knapp, warum das SD-Verfahren für konkave Polygone nicht funktioniert sowie warum das bei Scanline möglich ist.

Aufgabe 1

Im nebenstehenden Diagramm ist ein kleines Dreiecksnetz dargestellt. Angenommen für jeden Vertex werden die folgenden Attribute Position (3D) und Farbe (3D) mit jeweils `float`-Präzision (4 Byte pro Zahl) gespeichert.
(a) Wieviel Speicherplatz benötigt das Netz als sequenzielle Liste individueller Dreiecke?
(b) Listen Sie die Vertizes auf und erstellen Sie ein Indexed Face Set basierend auf den Indizes in diese Liste (wobei die Winding-Order Counter-Clockwise sein soll).
(c) Wieviel Speicherplatz benötigt das Indexed Face Set wenn die Indizes als `short` (2 Byte pro Zahl) dargestellt werden?



Aufgabe 2

Was ist die maximale Anzahl von Vertices, N_v, in einem IFS mit N_Δ Dreiecken?

Ausgehend davon, wieviel Speicher-Overhead kann durch ein IFS maximal (d.h. im worst case) auftreten?

Ein realistisches Szenario auf Basis eines Modells, das uns in späteren Übungen begegnen wird ist $N_\Delta = 262000$ und $N_v = 147000$, wobei Indizes als `int` verwendet werden und die Vertexdaten 32 Bytes benötigen. Wie ist der Speicherbedarf dieses Modells in Relation zur einfachen Dreiecksliste?

Aufgabe 3

Gegeben sei das Dreieck $\Delta_1 = ((0,0),(16,2),(12,4))$, bestimmen Sie mithilfe des Signed-Distance Verfahrens, ob sich die Punkte $\mathbf{p}_1 = (7,1)$, $\mathbf{p}_2 = (11,4)$ und $\mathbf{p}_3 = (13,3)$ innerhalb oder außerhalb von Δ_1 befinden.

Aufgabe 4

Angenommen auf Δ_1 soll eine Eigenschaft m der Oberfläche interpoliert werden, wobei die Werte an den Eckpunkten $m(\Delta_1) = \left(\frac{1}{2}, \frac{1}{3}, \frac{1}{4}\right)$ sind, wie lauten die baryzentrischen Koordinaten des Punktes p_3 aus der vorigen Aufgabe, und welchen Wert für $m(p_3)$ ergibt sich daraus?

Beispiele für m könnten z.B. der Rot-Kanal der Farbe sein, die Reflektivität der Oberfläche, der Grad der Verschmutzung, etc.

Aufgabe 5

In der Datei `raster-sd.asy` findet sich Asymptote-Code zum Rasterisieren von Dreiecken mittels des Signed-Distance Verfahrens. Vervollständigen Sie die Funktion `raster_signed_distance` und demonstrieren Sie die Funktionalität an einzelnen Beispielen die separate Pdf-Dateien ausgeben.

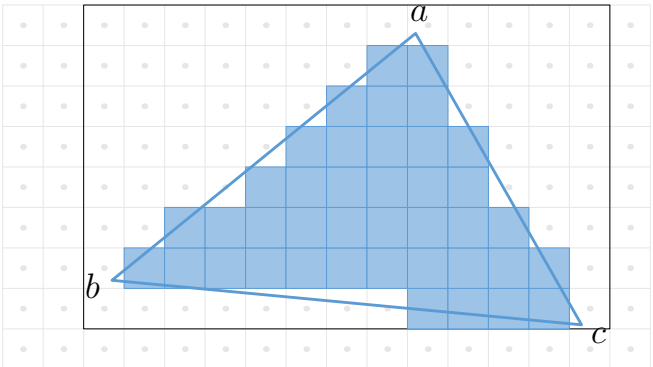
Initial die Bounding Box noch vernachlässigen und alle Pixel im Bild auf Überlappung mit dem Dreieck testen. Wenn diese Variante läuft behandeln Sie bitte den `use_bb` Parameter entsprechend und bestimmen Sie die Bounding Box und testen nur Pixel innerhalb dieser.

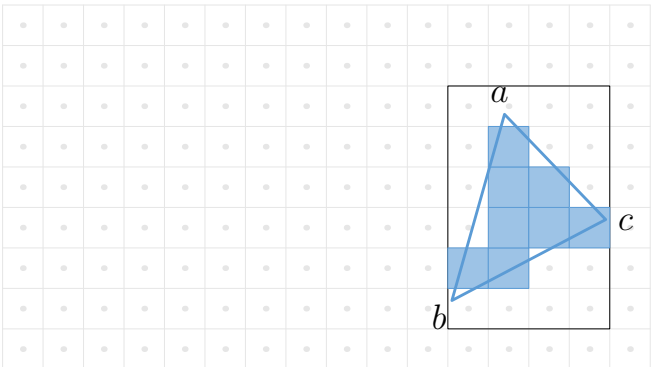
Vergleichen Sie die dann Performance beider Varianten auf Basis eines etwas höher aufgelösten Bildes mit verschieden großen Dreiecken.

Hinweise:

- Der Aufruf von Asymptote erfolgt via
  `shell$ asy -f pdf raster-sdf.asy`.
- Dokumentation zu Asymptote finden Sie unter
  → `https://asymptote.sourceforge.io/doc/`.
- Der Pixel $(0, 0)$ deckt die Fläche $(0, 0) \times (1, 1)$ ab, der Pixelmittelpunkt ist also $(\frac{1}{2}, \frac{1}{2})$.
- Es existiert schon eine Funktion `line_normal`, diese können Sie gerne lesen und verwenden.

Das Ergebnis sollte z.B. so aussehen:





(siehe nächste Seite)

Aufgabe 6

Für gewisse Anwendungen ist *konservative Rasterisierung*, d.h. das zeichen jedes Pixels, der ein Polygon auch nur irgendwie überlappt, nötig. Begründen Sie warum das mit auf Basis des Pixelmittelpunkts nicht klappt.

Erweitern Sie `raster_signed_distance` (s.o.) mit einem `bool conservative` parameter mittels dem konservativ rasterisiert werden kann. Falls dieser Parameter übergeben ist soll die Funktion für nicht getroffene Pixel prüfen, ob eine Dreieckskante den 'Rahmen' des Pixels schneidet. Es steht Ihnen frei Pfade und die Funktion `intersect` aus der Asymptote Pfad-Bibliothek zu verwenden.

Sind damit alle Fälle abgedeckt?

Das Ergebnis sollte analog zu oben (wobei die grün markieren Pixel durch konservatives Testen gefunden wurde) z.B. so aussehen: