Übungsblatt 6

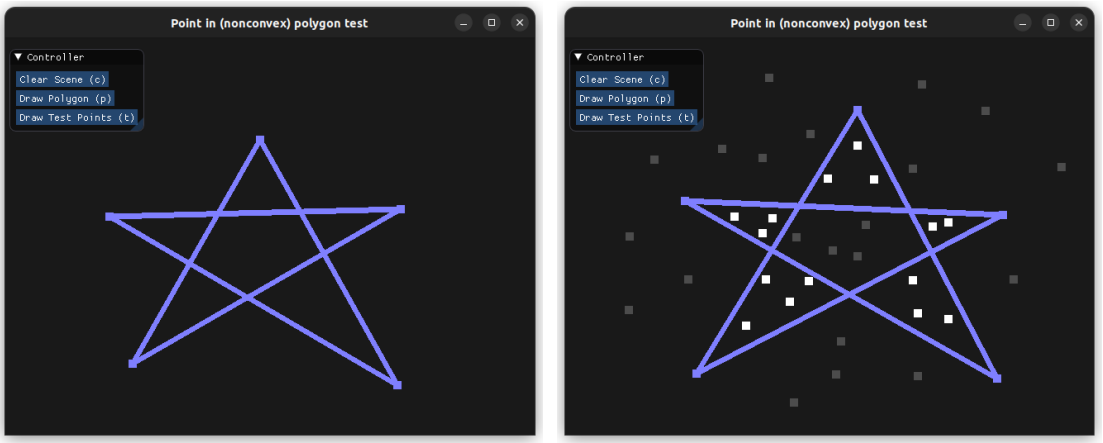
Willkommen zur sechsten Übung zur Vorlesung Generative Computergrafik. Dieses Blatt behandelt die Themen Clipping und Rasterisierung.

Aufgabe 1. Unter `cvmr.info/teaching/computergrafik/` finden Sie die Python Skripte `pointInPolygonTestTemplate.py` und `rendering_pointInPolygon.py`. Nach dem Starten des Programms können Sie mit Hilfe der Maus Punkte eingeben. Diese definieren solange ein Polygon, bis der Button `Draw Test Points` (Keyboard: '`t`') gedrückt wird. Die anschließend eingegebenen Punkte sollen dann darauf getestet werden, ob sie innerhalb oder ausserhalb des zuvor definierten Polygons liegen. Punkte die *innerhalb* des Polygons liegen sind *weiß*, Punkte *ausserhalb grau* darzustellen. Wird den Button `Draw Polygon` (Keyboard: '`p`') erneut gedrückt, soll das zuvor definierte Polygon erweitert werden können. Durch Klick auf den Button `Clear Scene` (Keyboard: '`c`') werden sowohl Punkte als auch das Polygon gelöscht.

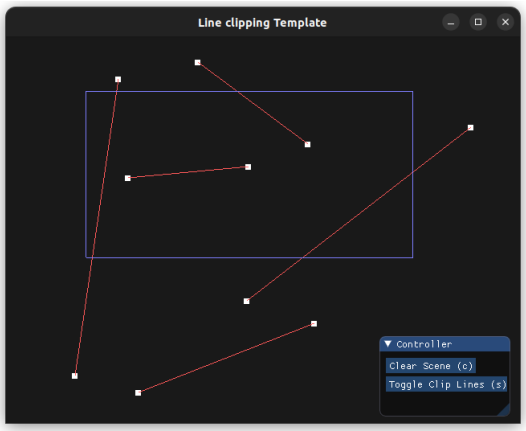
Um zu testen, wo ein Punkt $\mathbf{p}$ liegt, soll das Programm die, in der Vorlesung besprochene Beobachtung nutzen, dass ein beliebiger von $\mathbf{p}$ ausgehender Strahl genau dann eine ungerade Anzahl von Schnittpunkten mit dem Polygon aufweist, wenn $\mathbf{p}$ innerhalb des Polygons liegt.

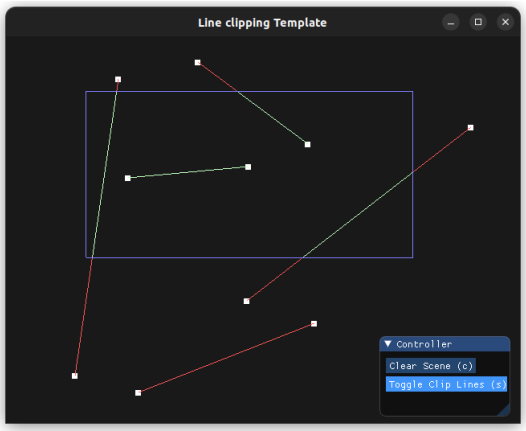
Implementieren Sie die Funktion `intersect(l1, l2)` in dem Python Programm `pointInPolygonTestTemplate.py`, welche überprüft, ob sich die beiden *Liniensegmente* `l1` und `l2` schneiden. Ein direkter Weg zur Implementierung dieser Funktion besteht darin, den Schnittpunkt der beiden, durch die Liniensegmente definierten Geraden zu berechnen und anschließend zu überprüfen, ob dieser Punkt zwischen den Endpunkten der beiden Strecken liegt. Ein weiterer Ansatz, welcher den *Drehsinn* dreier Punkte benutzt, findet sich zum Beispiel in dem Buch *Algorithmen und Datenstrukturen von R. Sedgewick.*

Die nachfolgenden Abbildungen zeigen ein Beispiel. Auf der linken Seite ist ein nicht konvexes Polygon zu sehen. Die rechte Seite entstand durch Eingabe weiterer Punkte. Wie zu erkennen ist, werden Punkte innerhalb des Polygons weiß dargestellt, während Punkte ausserhalb des Polygons grau gezeichnet werden.



Aufgabe 2. Unter `cvmr.info/teaching/computergrafik/` finden Sie die Python Skripte `lineClippingTemplate.py` und `rendering_lineClipping.py`. Das Programm erlaubt es, mit Hilfe der Maus Punkte zu setzen. Die ersten beiden Punkte definieren ein Rechteck. Von den weiteren definieren jeweils zwei eine Gerade. Implementieren Sie in diesem Programm die Funktionen `lineCode(...)` und `determine_clipped_line(...)` so, dass ein Lineclipping nach dem Algorithmus von Cohen-Sutherland durchgeführt wird. Die geclippten Linien sollen, wie in den nachfolgenden Abbildungen gezeigt, hell-grün dargestellt werden.





Aufgabe 3. Unter `cvmr.info/teaching/computergrafik/` finden Sie die Python Skripte `bresenhamTemplate.py` und `rendering_bresenham.py`. Das Programm erlaubt es mit Hilfe der Maus Punkte in einem Raster zu setzen. Je zwei Punkte werden dabei mit einer dünnen Linie verbunden (siehe Abbildung unten links). Implementieren Sie in dem Programm die Funktion `draw_Bresenham_line`. Diese soll mit Hilfe des Algorithmus von Bresenham die Kästchen (Pixel) des Gitters setzen, welche zu den jeweiligen Linien gehören. In der Abbildung unten rechts ist ein gewünschtes Ergebnis zu sehen. Beachten Sie, dass hier ein grause Kästchen einem Pixel entspricht (im Template beträgt die Pixelgröße standardmässig auf 20 (Canvaspixel) gesetzt).