Rastergraphik

Scheinkriterien: Für den Erhalt des ECG-Scheines ist es notwendig, insgesamt mindestens 37 Punkte zu erreichen. Zusätzlich gilt, dass pro Theorieblatt mindestens 1 Punkt und pro Praxisblatt mindestens 2 Punkte erreicht werden müssen!

3 Praxisaufgaben zu Grundlagen der Rasterisierung

In dieser Praxisaufgabe geht es um verschiedene Standardalgorithmen der Rastergraphik, wie sie heute verbreitete Anwendung finden. Die relevanten Algorithmen sollen dabei in Software nachimplementiert werden.

Hinweise zur Implementierung: Der Übung liegen verschiedene Materialien bei. Dabei handelt es sich um eine Musterlösung, in denen Sie die Ergebnisse der einzelnen Aufgaben betrachten können, ein lauffähiges Programm als Quelltext, dem die von Ihnen im Rahmen der Aufgaben zu erstellenden Algorithmen fehlen und eine allgemeine Erklärung zum Programmaufbau.

Die Bedienung der Musterlösung (und Ihres Arbeitsprogrammes) erfolgt mittels eines Kontextmenüs, das Sie über die rechte Maustaste erreichen. Neben den einzelnen Menüpunkten finden Sie in Klammern den entsprechenden Tastatur-Shortcut. Außerdem können Sie das Raster mit dem Mausrad zoomen und durch Gedrückthalten des Rades und Bewegen der Maus verschieben.

Eine Projektdatei für Visual-Studio 2010 finden Sie im Unterverzeichnis `build/vs2010`. Falls Sie die 2008er-Version verwenden finden Sie eine entsprechende Projektdatei im Unterverzeichnis `build/vs2008`. Die vorgegebenen Quelltexte sind ausführlich kommentiert und enthalten an den zu erweiternden Stellen zusätzliche Hinweise zur Lösung der Aufgaben. In der Implementierung existiert eine Oberklasse `abstract_tool`, welche die gemeinsamen Funktionalitöten aller Zeichenwerkzeuge kapselt. Diese besitzt mehrere Versionen der Methode `draw` zum Zeichnen von Objekten, die über einen, beziehungsweise zwei Punkte definiert sind.

3.1 Rasterisierung von Linien

In der Vorlesung wird auf spezielle Anforderungen an Algorithmen zur Rasterisierung von Linien eingegangen. In den folgenden Aufgaben sollen zwei etablierte Algorithmen, der DDA- und der Bresenham-Algorithmus, programmiert werden.

3.1.1 Digital Differential Analyzer (2Pt)

Implementieren Sie den DDA-Algorithmus zum Rastern von Linien, indem Sie die `draw`-Methode der Klasse `dda_line_tool` aus der Datei `dda_line_tool.cpp` vervollständigen. Stellen Sie dabei zunächst den Standardfall für das Linienrastern her.

3.1.2 Bresenham-Algorithmus (3Pt)

Implementieren Sie genauso den Bresenham-Algorithmus zum Rastern von Linien. Hierfür muss die `draw`-Methode der Klasse `bresenham_line_tool` aus der Datei `bresenham_line_tool.cpp` erweitert werden. Überlegen Sie für die Abgabe, welche Vor- und Nachteile der Bresenham-Algorithmus gegenüber dem DDA hat.

3.2 Füllalgorithmen

Zum Füllen existieren verschiedene Verfahren, die im Skript besprochen werden. Ihre Aufgabe besteht nun darin zwei grundlegende Füllalgorithmen nachzuvollziehen. Dabei handelt es sich zunächst um die naive Implementierung in Form eines rekursiven Algorithmus und anschließend um eine Erweiterung, welche ohne Rekursion auskommt.

3.2.1 Rekursives Füllen (1Pt)

Implementieren Sie den rekursiven Füllalgorithmus nach dem Skript in der `draw`-Methode der Klasse `recursive_fill_tool`. Erstellen Sie außerdem eine Routine, um sich die maximale Rekursionstiefe auf die Kommandozeile ausgeben zu lassen.

Hinweis: Die Funktion `toBeFilled(..)` aus dem Skript können Sie mit Hilfe von `canvas_store::get_pixel(..)` realisieren. Wenn das zu füllende Feld zu groß wird, erzeugt das rekursive Füllen einen Programmabsturz (stack overflow).

3.2.2 Nichtrekursives Füllen (4 Pt)

Implementieren Sie einen nichtrekursiven Füllalgorithmus, der auf dem FIFO-Prinzip (**f**irst **i**n, **f**irst **o**ut) beruht. Dabei werden die noch zu verarbeitenden Pixel in einem STL-Kontainer zwischengespeichert. Der Kontainer `std::deque` verfügt über die nötige Funktionalität. Mit `deque::push_back(..)` werden Elemente am Ende eingefügt. Mit `deque::front()` erhält man das vorderste Element des Kontainers und mit `deque::pop_front()` entfernt man es. Die Implementierung erfolgt in der Klasse `non_recursive_fill_tool`.

Hinweis: Der Programmabsturz aus Aufgabe 3.2.1 darf beim nichtrekursiven Füllen nicht mehr auftreten.

3.3 Zusatzaufgaben (insgesamt max.+5Pt)

- Implementieren Sie die Funktionalität des Rechteck-Werkzeuges um per Mausinteraktion ein Rechteck zu zeichnen. Das Grundgerüst für diese Aufgabe stellt die Klasse `rectangle_tool` dar, dessen Implementierung Sie in der Datei `rectangle_tool.cpp` finden (2Pt).
- Implementieren Sie einen Kreisrasterisierer nach Bresenham in `bresenham_circle_tool::draw(..)`! Recherchieren Sie selbständig, um die genaue Funktionsweise dieses Rasterisierers zu verstehen (5Pt).
- Implementieren sie den sog. Scanline-Fill-Algorithmus zum effizienten Füllen. Recherchieren Sie auch hier selbständig, um die genaue Funktionsweise des Scanline-Fills zu verstehen. Verwenden Sie die Testfüllform, um Ihr Ergebnis zu validieren. Die Implementierung soll in der `draw`-Methode der Klasse `line_fill_tool` erfolgen (5Pt).