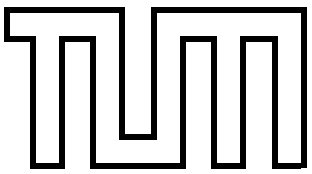


Technische Universität München

Fakultät für Informatik
Forschungs- und Lehreinheit Informatik VI

Übung zur Vorlesung Echtzeitsysteme

Einführung & Aufgabe 1-2

Philipp Heise
heise@in.tum.de

Steffen Wittmeier
steffen.wittmeier@in.tum.de

Christoph Staub
staub@in.tum.de

Michael Jäntsch
michael.jaentsch@in.tum.de

Wintersemester 2012/13

Aufgabe 1: Pointer und Speicherallokation

Zum tieferen Verständnis von Pointern und Speicherallokation sollen sie einige einfache Operationen auf Graustufen-Bildern implementieren. Im Verzeichnis `pgm` finden sie ein Rahmenprogramm um *PGM*-Bilder zu lesen und zu schreiben. Die benötigten Methoden sind

```
void PGM::write( const char* filename, unsigned char* ptr, int w, int h )
```

`filename` gibt dabei die Zieldatei zum schreiben an. `ptr` zeigt auf einen Speicherbereich an dem sich $w \times h$ Graustufen-Pixel des Bildes befinden.

```
void PGM::read( const char* filename, unsigned char** ptr, int* w, int* h )
```

`filename` gibt dabei die Quelldatei zum lesen an. `ptr` ist die Adresse eines Pointer an dem sich nach dem Aufruf die Pixel-Daten des Bildes befinden. `w` und `h` sind jeweils Pointer auf Integers, die nach dem Aufruf die Breite und Höhe der Bilddatei enthalten.

PGM-Bilder können sie mit einem geeigneten Anzeigeprogramm unter Linux z.B. GIMP betrachten.

Aufgabe: Helligkeit und Kontrast

Die Funktion `brightnessContrast` in der Datei `bc.cpp` soll jedes unsigned char Pixel mit einem Wertebereich von $0 \ldots 255$ lesen und in einen float-Wert im Bereich $0 \ldots 1$ umwandeln. Der resultierende `float` Wert soll dann mit dem Kontrastparameter c multipliziert werden und der Helligkeitsparameter b addiert werden. Anschließend soll der berechnete Wert wieder auf den Bereich $0 \ldots 1$ begrenzt werden und in einen `unsigned char` ($0 \ldots 255$) umgewandelt werden und im Bild zurückgespeichert werden. Pseudocode:

```
Lese Pixelwert
Konvertierung nach float-Wert f
f = f * c + b
Begrenzung von f in den Bereich 0 bis 1
Konvertierung von f in einen unsigned char
Speichere neu berechneten Wert
```

Hinweise:

- `unsigned char` c im Bereich 0 bis 255 zu float f ($0 \ldots 1$):
  `f = ( ( float ) c ) / 255.0f`
- `float` f im Bereich 0 bis 1 zu unsigned char c:
  `c = ( unsigned char ) ( f * 255.0f )`

- Alle `unsigned char` Werte liegen hintereinander im Speicher und können unabhängig von ihrer Bildposition bearbeitet werden

Aufgabe: Bilderzeugung

Implementieren sie die Funktion `filledCircle` in der Datei `circle.cpp`, die ein Graustufenbild der Größe $w \times h$ mit einem gefüllten Kreis zurückgibt. Die Parameter `cx` und `cy` bestimmen den Mittelpunkt des Kreises und `radius` den Radius. Um festzustellen ob ein Pixel an der Stelle x, y gefüllt werden muss nutzen sie die folgende Berechnungsmethode

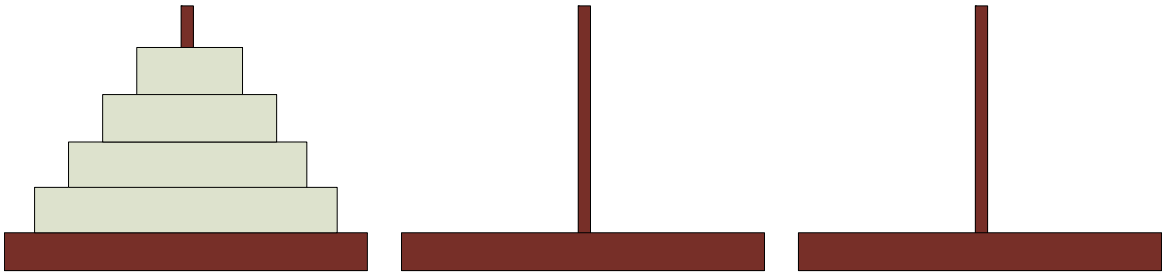
$$f(x, y, cx, cy, radius) = \begin{cases} 255 & \text{if } \sqrt{(x-cx)^2 + (y-cy)^2} \leq radius \\ 0 & \text{if } \sqrt{(x-cx)^2 + (y-cy)^2} > radius \end{cases} \quad (1)$$

Hinweise:

- Bei einer Bildgröße von $w \times h$ können sie die lineare Speicherposition des Pixels x, y mit $y * w + x$ berechnen
- Um die Formel auszuwerten wandeln sie x, y und die notwendigen Parameter in `float` Werte um
- Zur Speicherallokation nutzen sie `new`
- Alle `unsigned char` Werte liegen hintereinander im Speicher und können unabhängig von ihrer Bildposition bearbeitet werden

Aufgabe 2: Türme von Hanoi

Ziel



Die Türme von Hanoi sind ein mathematisches Knobel- und Geduldspiel bei dem ein Turm der Höhe n von der Stelle A nach Stelle C (mit Ablagestelle B) so transportiert werden, dass man immer nur eine Scheibe nehmen kann und niemals eine größere Scheibe über einer kleineren liegt. Ziel dieser Aufgabe ist es, die Kenntnisse der Programmiersprache C++ zu vertiefen.

Aufgabe

Sie finden ein Rahmenprogramm für diese Aufgabe im Verzeichnis `hanoi`. Implementieren sie die Methode `Hanoi::solveRecursive`. Nutzen sie dazu die Methoden `Rod::pop` und `Rod::push`, die jeweils eine Scheibe von einem Stab entfernen bzw. hinzufügen.
Verwenden Sie zur Implementierung den nachfolgend im Pseudocode dargestellten Algorithmus.

```
funktion bewege (Stab a, Stab b, Stab c, Zahl i ) {
    falls (i > 0) {
       bewege(i-1, a, c, b);
       verschiebe oberste Scheibe von a nach c;
       // evtl. Ausgabe
       bewege(i-1, b, a, c);
    }
}
```

Nutzen sie die Methode `Hanoi::print` nach jeder Scheibenbewegung, um den momentanen Zustand anzuzeigen. Testen Sie Ihr Programm für mehrere Turmhöhen durch Modifikation des Konstruktor-Parameters der Klasse `Hanoi` in `main.cpp`.