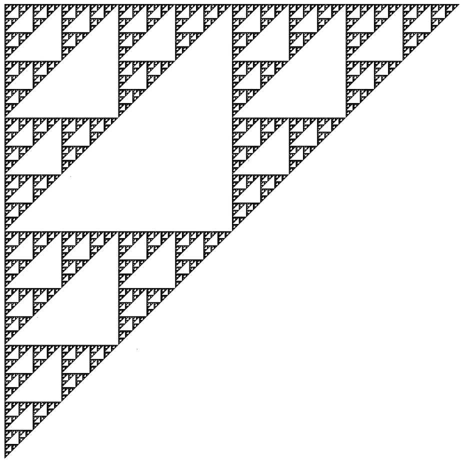
Allgemeine Informatik III / Systemnahe Software I (WS 2008)

Abgabetermin: 09.12.2008

1 Aufgabe 1 (10 Punkte)

In dieser Aufgabe sollen Sie ein Sierpinski-Dreieck, wie in Abbildung 1 dargestellt, zeichnen. Ein Sierpinski-Dreieck entsteht durch wiederholte Aufteilung eines Dreiecks in vier kongruente Dreiecke oder durch ein Iteriertes Funktionensystem welches im nächsten Abschnitt vorgestellt wird.



Ein Sierpinski-Dreieck wie es in dieser Aufgabe erzeugt werden soll.

1.1 Beschreibung

Die Berechnung der Punkte ($p[x, y]$) dieses Bildes erfolgt durch folgende Funktionen

```
/*
 * p[0] entspricht x
 * p[1] entspricht y
*/
void sierpinski_1(float p[2]) {
  p[0] /= 2;
  p[1] /= 2;
}
void sierpinski_2(float p[2]) {
  p[0] = (p[0]+1)/2;
  p[1] = p[1]/2;
}
void sierpinski_3(float p[2]) {
  p[0] = p[0]/2;
  p[1] = (p[1]+1)/2;
}
```

Zuerst wird ein **zufälliger** Startpunkt $p[x_0, y_0]$, mit $x_0, y_0 \in [0, 1]$, festgelegt. Dann wird **zufällig** eine der 3 Funktionen ausgewählt und angewendet, wir erhalten den Punkt $p[x_1, y_1]$ und zeichnen ihn. Nun wählen wir wieder zufällig eine der 3 Funktionen aus und wenden sie auf den Punkt $p[x_1, y_1]$ an. So erhalten wir Punkt $p[x_2, y_2]$ und zeichnen ihn. Dieser Vorgang wird mehrere Tausend mal wiederholt.

1.2 Aufgabenstellung

Schreiben Sie ein Programm welches ein Sierpinski-Dreieck erzeugt. Hierzu lesen Sie die Anzahl der Iterationen *unsigned int num_iterations*, die Breite des Bildes *unsigned int width* und die Höhe des Bildes *unsigned int height* ein. Das Bild soll in einer dynamischen $height \times width$ großen *char*-Matrix gespeichert werden. Zuerst müssen alle Werte dieser Matrix mit 255 (Weiß) initialisiert werden. Nun können die Punkte, mit Hilfe der oben beschrieben Funktionen, berechnet werden. Achten Sie darauf, dass die Werte, welche den Funktionen übergeben werden, zwischen 0 und 1 liegen müssen!

Um die berechneten Punkte ($p[x, y] \in [0, 1]$) in der Matrix einzutragen, müssen diese zuvor noch auf *height* bzw. *width* skaliert werden:

```
unsigned int i = (unsigned int)(p[0] * width);
unsigned int j = (unsigned int)(p[1] * height);
```

Um nun den Punkt an der Stelle i, j zu zeichnen, wird der Wert an dieser Stelle auf 0 (Schwarz) gesetzt.

Wurden *num_iterations* Punkte berechnet, übernimmt die Funktion *writePGM* die Ausgabe der Matrix als Portable Graymap (PGM). Diese erwartet als Parameter einen Namen für das Bild, einen Zeiger auf die Matrix, deren Höhe und Breite.

```
void writePGM(const char* filename, char *bild,
              unsigned int width, unsigned int height) {

  FILE* fp = fopen(filename, "wb");
  fprintf(fp, "P5\n%u\n%u\n255\n", width, height);
```

```
  fwrite(bild, width*height, 1, fp);
  fclose(fp);
}

//Beispielaufruf:
writePGM("sierpinski.pgm", bild, width, height);
```

Das erzeugt Bild lässt sich unter Linux/Unix z.B. mit gimp oder xv betrachten. Um die Bilder unter Windows anzusehen bietet sich das kostenlose IrfanView an.

1.3 Hinweis

Ganzzahlige Zufallszahlen erhält man in C durch die Funktion *rand*(). Hieraus kann man leicht Gleitkommazahlen zwischen 0 und 1 erzeugen.

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int main(void) {
   int i, z;
   float f;
   //Zufallszahlengenerator initialisieren. Nur 1 mal erforderlich!
   srand(time(NULL));
   for(i = 0; i < 10; i++) {
      z = rand();
      f = (float)z / (float) RAND_MAX;
      printf("Die Zufallszahl lautet %d\n", z );
      printf("Und als Gleitkommazahl zwischen 0 und 1: %f\n", f );
   }
   return 0;
}
```

Viel Erfolg!