

## Aufgabe 10

Dennis Blöte, 08.02.2007

### Rasterbilder

Bei der gegebenen Aufgabenstellung kann es insgesamt  $2^{(a/n)*2}$  verschiedene Bilder geben. Jedes der Felder kann entweder schwarz oder weiss sein - daraus ergibt sich, dass es 2 hoch die Anzahl der Felder verschiedene Möglichkeiten gibt. Aus dem Umstand, dass es sich um binäre Zustände handelt, resultierte dann auch mein Lösungsansatz zur systematischen Erzeugung der einzelnen Bilder:

Alles was man machen muss, um jeden Zustand zu generieren ist, einen Zähler zu haben und diesen nach jedem gezeichneten Bild zu erhöhen. Den Wert dieses Zählers wandelt man vor dem Zeichnen des Bildes mit der Funktion `binary()` in einen String, welcher den Wert in binärer Form abbildet. Das Bild 100 (Zählerstand = 99, da wir bei 0 anfangen) entspricht somit beispielsweise dem Binärstring „1100011“.

Beim Zeichnen des Bildes benutze ich diesen String, um die Farbwerte der einzelnen Felder zu bestimmen - siehe die Funktion `get_value()`:

```
int get_value(int index)
{
    int len = state.length();
    int g = len - index;
    String sub = (index <= len) ? state.substring(g, g+1) : "0";
    return sub.equals("1") ? 0 : 255;
}
```

Ihr wird der Index des Felds übergeben, dessen Farbwert die Funktion zurückgeben soll (das erste Feld hat den Index 1). Dieser Index wird nun genutzt, um den String vom Ende an abzufragen - Index 1 würde so auf das letzte Zeichen, 2 auf das vorletzte Zeichen des Strings zugreifen.

```
Index:   7 6 5 4 3 2 1
String: „1 1 0 0 0 1 1“
```

Ist der Wert des abgefragten Zeichens 1, so wird der Farbwert 0 (schwarz) zurückgegeben, ansonsten 255 (weiss).

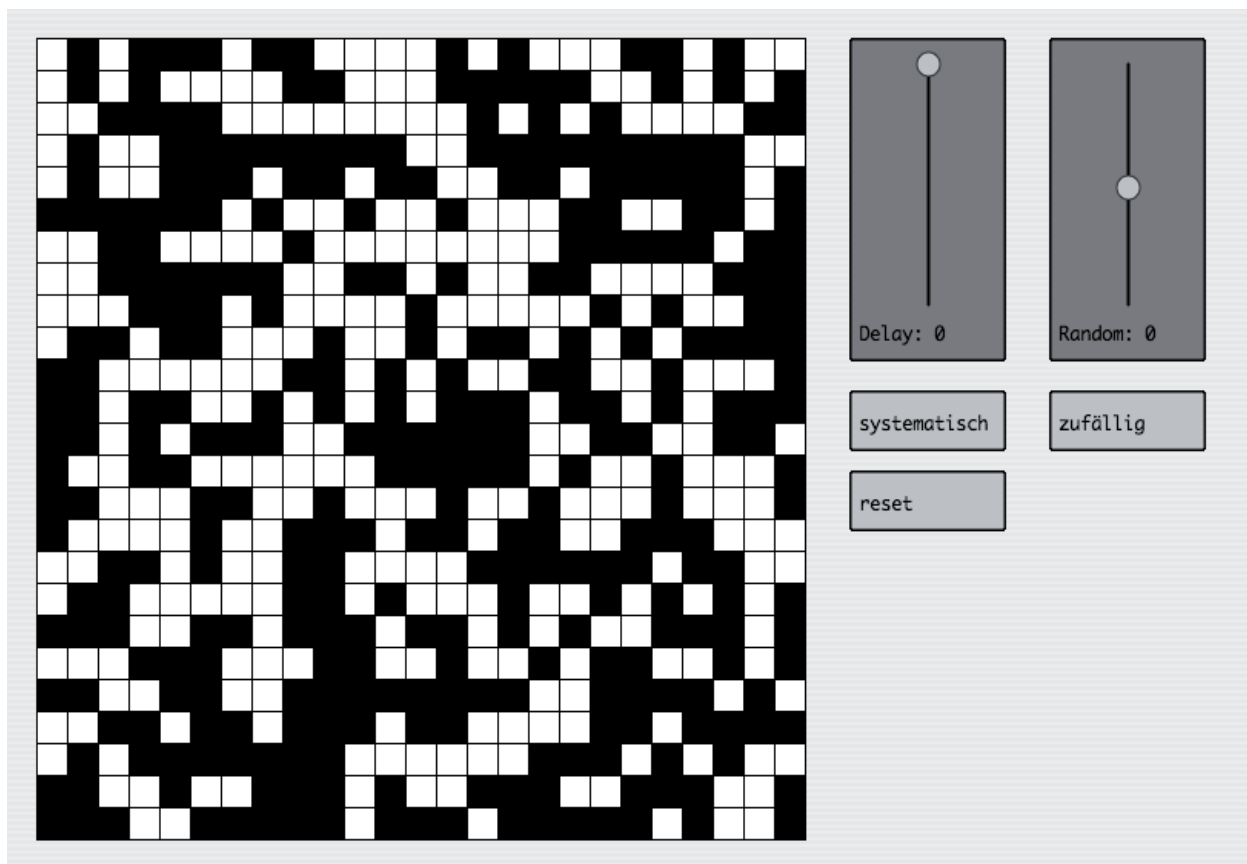
Auch wenn diese Lösung auf den ersten Blick etwas kompliziert erscheinen mag, so war sie für mich nach langem Nachdenken darüber die sinnvollste, weil man so jeden Zustand des Bildes einfach generieren kann, ohne dass sich Doppelungen ergeben (der Counter wird nach jedem Zeichenvorgang erhöht, somit gibt es jeden Zustand nur einmal).

Durch die Schieberegler kann man wie in der Aufgabenstellung vorgegebenen Einfluß auf die Bildgenerierung nehmen. Der Regler Delay setzt die Verzögerung nach jedem Zeichenvorgang in Millisekunden. Dazu wird der eingestellte Wert von der Processing-Funktion `delay()` übernommen, welche die Verzögerung verursacht. Maximalwert ist in diesem Fall eine Sekunde (1000 ms).

Die Verzögerung wirkt sich nur aus, wenn der Modus systematisches Zeichnen aktiviert ist - dies erreicht man durch das Drücken des Buttons „systematisch“.

Der zweite Schieberegler beeinflusst die Generierung der (Pseudo-) Zufallsbilder. Sein Wert kann zwischen -4 und 4 eingestellt werden, wobei -4 größtenteils weisse und 4 hauptsächlich schwarze Rechtecke bewirken.

Ist der Zufallsmodus aktiviert (durch Drücken des Buttons „zufällig“), wird nach jedem Mausklick ein neues Bild mit dem eingestellten Zufallswert erzeugt.



Der Screenshot zeigt das Programm im Zufallsmodus mit gleichmäßiger Verteilung der schwarzen und weissen Rechtecke (Zufallswert = 0).

Für die zufällige Generierung der Farbwerte wird wie beim systematischen Ablauf ein String auf Einsen und Nullen erzeugt. Dafür wird in einer Schleife für jedes der Felder (COUNT\_FIELDS) ein Zufallswert zwischen 1 und 0 berechnet, welcher durch den im Schieberegler Random eingestellten Wert beeinflusst wird (seed). Da der so zustande kommende Wert größer als 1 oder kleiner als 0 sein kann, wird er gegebenenfalls angepasst. Anschließend wird die erzeugte 1 oder 0 an den String state angefügt.

```
void draw_random()
{
    state = "";
    // Erzeugen eines Strings aus 0 un
    for(int i=0; i<COUNT_FIELDS; i++) {
        // Einstellung bei Wertberechnung mit einbeziehen
        float seed = float(ctrl_rand.value)/10;
        int k = round(random(1) + seed);
        // Wert ggf. auf 0 oder 1 anpassen
        k = k<0 ? 0 : k;
        k = k>1 ? 1 : k;
        state += k;
    }
    draw_rects();
}
```

Da Processing von Haus aus leider keine Bibliothek zur Erzeugung von Kontrollelementen mitbringt, habe ich die Buttons und Schieberegler selbst erstellt.

Dafür habe ich die Klassen Button und Slider geschrieben, welche so modular sein sollten, dass man sie auch in anderen Programmen einsetzen könnte. Das Programmieren einzelner Klassen bot sich in diesem Fall an, weil man für die Realisierung der Aufgabe mehrere Instanzen der Kontrollelemente benötigt und der Programmcode so modularer aufgebaut ist.

Ich hoffe ich habe den Code verständlich kommentiert, so dass man beim Lesen einigermaßen meinen Gedanken folgen kann. Die Stellen, welche ich für weiter erklärungsbedürftig halte, habe ich hier näher beschrieben. Meine Realisierung der Aufgabenstellung umfasst somit die Aufgabenteile a) bis d) - für den Rest ist vorerst keine Zeit.