

random01

Jörn Ketelsen

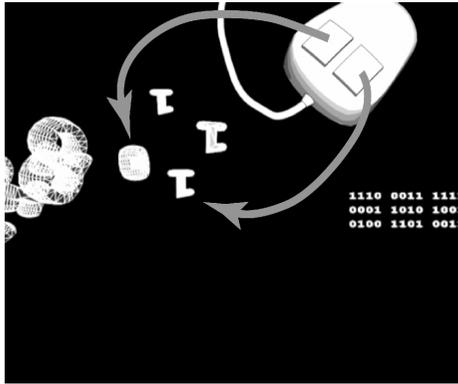


Abb.1: Bitingabe: Die linke Maustaste erzeugt Einsen, die rechte Nullen.

Im Rahmen des Projektes Illusion&Interface entstand Software, die wir „Lernlabore“ nennen. Das sind Umgebungen, in denen Benutzende durch Experimentieren Bilder erzeugen können, um Einblicke in algorithmische Grundlagen der Computerkunst zu erlangen.

Thema des Labors random01 sollte es nicht sein, ästhetisch wertvolle Bilder zu erzeugen, sondern vielmehr, ein einfaches, aber zunächst verwirrendes Interface bereitzustellen, welches sich per trial-and-error und geeignete grafische Präsentation dem Benutzenden erklärt.

Wenn sich dabei Erkenntnisse über Zufall und Datenrepräsentation einstellen sollten, so ist dies wünschenswert.

Eingabe

Beim Start des Labors öffnet sich zunächst ein schwarzer Bildschirm.

Nach kurzer Zeit wird der Benutzende wahrscheinlich versuchen, per Maus eine Veränderung herbeizuführen. Wenn diese Veränderung nicht darin besteht, das Fenster schließen zu wollen, sondern innerhalb des Fensters durch Klick gesucht wird, erscheint ein Zeichen, das auf einen bestimmten Punkt auf dem Bildschirm zustürzt. Auf dem Weg dorthin wandelt sich seine Darstellung von

einem Drahtgittermodell zu einer solchen mit undurchsichtiger weißer Oberfläche.

Sollte der Benutzende versuchen, den Vorgang zu wiederholen, wird er schließlich feststellen, dass jeder Klick ein neues Zeichen erzeugt, wobei dieses entweder eine Eins oder eine Null ist. Diese Zeichen ordnen sich in Zeilen und Spalten an.

Absicht bei dieser Lernumgebung war es, eine möglichst einfache, obschon verwirrende Benutzung zu schaffen (also ein Interface), um dennoch eine nicht ganz so einfache, wenn auch reduzierte Grafik zu beschreiben.

Weder die Tastatur sollte Benutzung finden, noch die üblichen Pull-Down und Pop-Up Menüs. Mit diesen Einschränkungen stehen dem Benutzenden am PC im Grunde nur noch die beiden Maustasten zur Verfügung.

Nun könnte man auf die Idee kommen, da der Computer ohnehin digital arbeitet, die beiden Maustasten ebenfalls digital zu interpretieren, nämlich eine Taste als 0, die andere als 1.

Sollten die Folgen von Nullen und Einsen nun die Operationen des Computers steuern, so ergab sich eine gewaltige zeitliche Differenz: Während der Computer Folgen von 0 und 1 nahezu beliebig schnell erzeugen

kann, ist der Mensch weder in der Lage, noch bereit, dies zu tun.

Andererseits sollten die eingegebenen Binärmuster die Grundlage für die Erzeugung eines Bildes liefern, derart, dass eine vom Benutzer erzeugte binäre Folge vom Computer als Codierung eines Bildes interpretiert wird.

Diese Erzeugung eines Bildes aus einer Folge von Nullen und Einsen kann auf sinnvolle Art und Weise nur geschehen, wenn die Länge der Folge ausreichend groß ist.

Es musste also ein Mittelweg zwischen langsamer menschlicher Eingabe und schneller maschineller Ausführung gefunden werden.

Dazu findet zunächst eine Trennung zwischen Eingabe und Ausführung statt. Beide Teile entfalten bewegte Bildlichkeit auf dem Bildschirm und beanspruchen dadurch einige Aufmerksamkeit.

Die Anzahl von 64 Nullen oder Einsen schien geeignet zu sein. Zum einen ist die Mühe, 64 Zeichen per Maus einzugeben, nicht allzu hoch; zum anderen lassen diese Zeichen sich als 8 Byte interpretieren, was vielleicht im Folgenden Vorteile bringt.

Außerdem lassen sie sich grafisch kompakt darstellen, beispielsweise als 8-mal-8 Bit Block, oder als 4 Zeilen mit je vier Nibbles.

Die letzte Möglichkeit wurde tatsächlich gewählt, um die Zusammengehörigkeit von 4 Bits zu einem Nibble zu verdeutlichen.

Der Benutzer gibt 64 Bits b_i ein. Aus je 4 dieser Bits ergibt sich, bedingt durch die Reihenfolge der Eingabe, eine Folge von Nibbles n_i , wobei $n_0 = b_0b_1b_2b_3$, $n_1 = b_4b_5b_6b_7, \dots, n_{15} = b_{60}b_{61}b_{62}b_{63}$. Die Folge $N = \{n_0, n_1, \dots, n_{15}\}$ dieser Nibbles soll einen elementaren Zeichenvorrat bereitstellen, aus dem im nächsten Schritt Befehle zusammengesetzt werden.

Zufall

Nachdem die Eingabe abgeschlossen ist, beginnen sich die Nibbles nach einiger Zeit in scheinbar zufälliger Reihenfolge Stück für Stück in den oberen, linken Bereich des Bildschirms zu bewegen, um sich dort zeilenweise anzuordnen.

Dabei wandelt sich wieder die Darstellung der Zeichen. Sie erscheinen nun als schattierte, körperhafte Objekte¹.

Der Name des Prototypen deutet es an: Bei diesem Vorgang sind zufällige Elemente beteiligt.

Die Eingabe wird vermutlich schon eine

¹ Dieser schrittweise Übergang zu mehr „Körperlichkeit“ soll der Aufladung der Zeichen mit Bedeutung entsprechen.

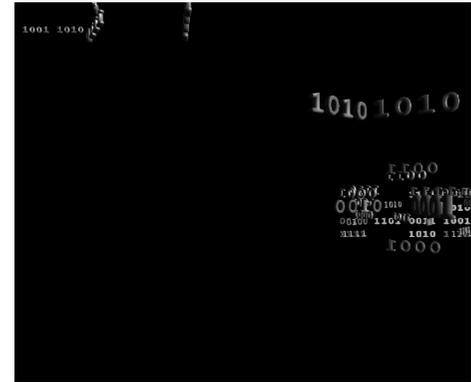


Abb.2: Die Eingabe ist abgeschlossen, die Nibbles ordnen sich zufällig neu an.

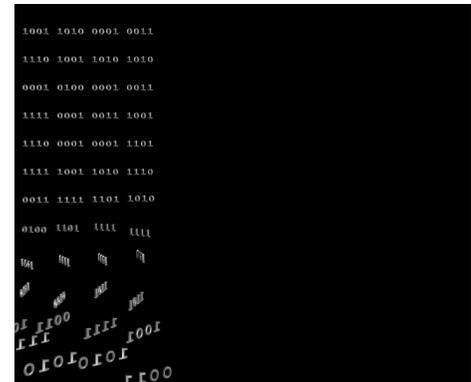


Abb.3: Die Neuordnung der Nibbles ist fast abgeschlossen, die 12 Befehlszeilen sind schon erkennbar.

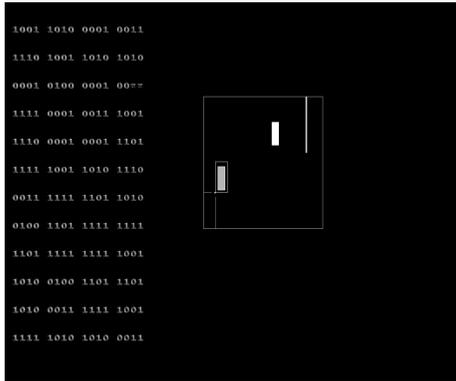


Abb.4: Die Befehle werden verarbeitet, aktuell die letzten beiden Bits der dritten Zeile, eine Farbangabe für das dritte Bildelement.

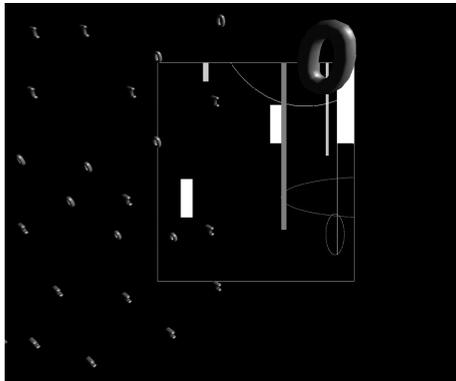


Abb.5: Auf dem Weg zur vergrößerten Bildarstellung. Die Zeichen machen Platz für das Bild.

zufällige Bitfolge liefern, da der Benutzer zunächst nicht wissen wird, welche Bedeutung hinter welchem Zeichen steht.

Bei wiederholtem Gebrauch des Labores wird er schließlich doch Bedeutungen hinter den Zeichen erkennen können. Damit wird man zwar selten in die Lage kommen, das Aussehen des späteren Bildes genau zu wissen, aber zumindest beeinflussen² kann man es.

Zusätzlich zur oben genannten, durch die manuelle Eingabe bedingten Unordnung wählt der Computer pseudozufällig jeweils eins der eingegebenen Nibbles aus.

Dieses selektierte Nibble wird nun in den Befehlsbereich bewegt, um dort an die jeweils aktuelle Zeile angefügt zu werden. Ein Nibble aus dem Eingabebereich kann dabei dreimal selektiert werden. Dies dient dazu, aus den tatsächlich eingegebenen 8 Byte einen Code von insgesamt 24 Byte Länge zu erzeugen.

Hier wird auch der Grund klar, warum die Nibbles als kleinste bedeutungstragende Elemente gewählt wurden: Hätten dafür Bits Verwendung gefunden, wäre die Bedeutung

der Reihenfolge der Eingabe (also der Einfluss des Benutzers) durch das zufällige Verstreuen verloren gegangen.

Das Verstreuen der 64 eingegebenen Bits hätte zu 64! verschiedenen Anordnungen oder Mustern geführt. Hätte man umgekehrt größere Einheiten, zum Beispiel Bytes gewählt, so wäre der direkte Einfluss des Benutzers zu groß geworden, um den Prototypen noch mit Zufall in Verbindung zu bringen. Zum einen wäre der Benutzer in der Lage gewesen, die Bytes quasi zu programmieren, sobald er ihre Bedeutung durchschaut hätte, zum anderen wären nur 8! Möglichkeiten der Kombination der eingegebenen Bytes geblieben.

Es wurde also der Mittelweg der Nibbles gewählt. Hinter ihnen lässt sich zum einen noch genug Bedeutung verbergen; zum anderen ist die Zahl der Kombinationen noch recht hoch.

Diese Anzahl wollen wir berechnen: Der Reihe nach werden die Nibbles jeweils dreimal selektiert und dann an eine von 48 Positionen bewegt.

Dies bedeutet für das erste Nibble bei der ersten Selektion 48 mögliche Positionen, bei der zweiten 47 und bei der dritten 46. Die Anzahl der Möglichkeiten, diese drei identischen Nibbles auf drei Positionen aus 48

² Einschränkung sei allerdings gesagt, dass der Benutzende das Aussehen des Ergebnisses in trivialen Fällen tatsächlich explizit erzwingen kann, zum Beispiel im Fall $b_0=b_1=b_2=...=b_{63}$.

zu verteilen, ist also 48·47·46. Da die drei Nibbles aber identisch sind, ist es egal, welches an welcher der drei Positionen steht.

Die Zahl der Kombinationen muss also nach unten korrigiert werden. Damit erhält man für das erste Nibble:

$$\frac{48 \cdot 47 \cdot 46}{3!}$$

Es folgt damit für die gesamte Anzahl:

$$C_{48} = \frac{48 \cdot 47 \cdot 46}{3!} \cdot \frac{45 \cdot 44 \cdot 43}{3!} \cdot \dots \cdot \frac{3 \cdot 2 \cdot 1}{3!} = \frac{48!}{3!^{16}}$$

Dieser Ausdruck gilt für den Fall, dass alle Nibbles verschieden sind und lässt sich mit Hilfe der Stirling Approximation annähern:

$$C_{48} \approx \left(\frac{48}{e}\right)^{48} \cdot \frac{\sqrt{2\pi \cdot 48}}{10^{16 \cdot \log 6}} \approx 4,4 \cdot 10^{48}$$

Eine gewaltige Anzahl von Kombinationen also, von denen viele allerdings kaum unterscheidbar sind.

Die Anordnung im genannten Befehlsbereich erfolgt in Zeilen à vier Nibbles. Eine solche Zeile wird im Folgenden als Befehl interpretiert. Die Generierung dieser Befehle terminiert, wenn zwölf von ihnen vorhanden sind, und damit der Vorrat der Nibbles erschöpft ist.

Interpretation

Nachdem alle Nibbles sich im oberen, linken Bereich befinden, erscheint rechts davon

ein Rechteck, die zu füllende Bildfläche. Hier wird stückweise das Bild aufgebaut. Die Binärmuster im Befehlsbereich werden als eine Art primitiver Maschinencode aufgefasst. Er soll einerseits komplex genug sein, um unterscheidbare Bilder zu erzeugen, andererseits so simpel, dass man ihn noch nachvollziehen kann.

In den 16 Bit bzw. 4 Nibbles jedes Befehls finden sich Angaben zur Art des Elements, Farbe, Größe und Position. Es gibt vier Arten von Elementen: ein gefülltes Rechteck, ein nicht gefüllter Kreis, eine horizontale Linie, eine vertikale Linie.

Zur Färbung stehen vier Abstufungen von Grau zur Verfügung. Die Positionierung der Elemente erfolgt über die ersten beiden Nibbles jedes Befehls. Die dort angegebene Zahl wird mit 10 multipliziert, um vernünftige Bildschirmkoordinaten zu bekommen. Es ergibt sich das Problem, dass mit einem Nibble nur 16 Adressierungen möglich sind. Das erschien etwas wenig. Das Problem wurde damit umgangen, dass am oberen und am rechten Bildrand jeweils fünf zusätzliche Zeilen beziehungsweise Spalten in die Bildfläche eingefügt wurden. Diese Ränder können nur durch Überlängen der grafischen Elemente beschrieben werden, nicht durch eine direkte Positionierung.

Bitnummer	Bedeutung	Werte
0..3	x-Position	0000: 0 ... 1111: 15
4..7	y-Position	0000: 0 ... 1111: 15
8..9	Breite	00: 0 ... 11: 3
10..11	Höhe	00: 1 ... 11: 3
12..13	Elementtyp	00: gefülltes Rechteck 01: horizontale Linie 10: vertikale Linie 11: ungefüllter Kreis
14..15	Farbe	00: rgb=52,52,52 01: rgb=127,127,127 10: rgb=204,204,204 11: rgb=255,255,255

Tab.1: Die Bedeutung der Bits eines Befehls in Abhängigkeit von ihrer Position im Befehl.

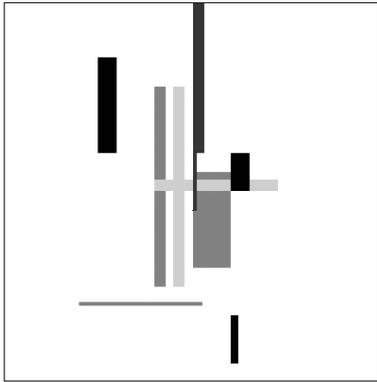


Abb.6: Ein mit dem Labor erzeugtes Bild in Falschfarbendarstellung.

Die Größe eines grafischen Elementes wird im dritten Nibble festgelegt, wobei die Höhe in den ersten beiden, die Breite in den beiden letzten Bits steht. Auch diese Zahl wird wieder verzehnfacht.

Während des Bildaufbaus rotiert die Bitgruppe, die gerade interpretiert wird, um ihre Längsachse. Handelt es sich dabei um eine Angabe der Position, so bewegt sich der Cursor auf der Bildfläche zunächst zum Mittelpunkt des Bildes und von dort zur angegebenen Position.

Handelt es sich um eine Größenangabe, so wird ausgehend von der aktuellen Position des Cursors ein Rahmen nach rechts und oben aufgezogen, der der angegebenen Größe entspricht. Bei einer Angabe über die Art des Elements wird im bereits aufgezogenen Rahmen ein Umriss des jeweiligen Elements erzeugt und dieser dann bei einer folgenden Farbangeabe gefärbt bzw. farbig gefüllt.

Ausgabe

Sobald das Bild erschienen ist, kann man es vergrößern, indem man auf die linke Maustaste drückt. Die rechte führt zu einem neuen Eingabebildschirm.

Ist das Bild vergrößert, speichert ein weiterer Druck auf die linke Maustaste das Bild

ab, ein Druck auf die rechte beendet die Vergrößerung. Durchgängig wurde versucht, die linke Maustaste für die Bedeutung „an“, die rechte für „aus“ zu verwenden.

Das Bild wird im PNG-Format gespeichert und automatisch in ein HTML-Dokument eingefügt, das der Reihe nach die gespeicherten Bilder aufnimmt, und so eine einfache Galerie³ realisiert. Das PNG-Format wurde gewählt, weil die Bilder im WWW darstellbar sein sollten. Damit kamen im wesentlichen nur JPEG, GIF und eben PNG in Frage. JPEG ist für Bilder, wie sie hier erzeugt werden, eher ungeeignet. Die Bibliotheken für das GIF-Format hingegen sind noch lizenzpflichtig. blieb also PNG übrig.

Der Umfang der Galerie ist nur von der Geduld des Benutzers abhängig. Theoretisch sind 2^{64} , also ungefähr 10^{20} verschiedene Bilder möglich. Diese sind allerdings in vielen Fällen gar nicht oder kaum unterscheidbar. Aber selbst wenn man je 100000 Bilder zu einer Klasse zusammenfasst, innerhalb der die Bilder nicht unterscheidbar sind, bleiben immer noch 10^{15} , also Millionen Milliarden Möglichkeiten.

Festzuhalten bleibt also: Der Rechner

³ Ein Beispiel einer solchen Galerie findet sich unter <http://www.ieti.de/interna/prototypen/random01/example/random01.html>.

erhält in diesem Prototypen eine Menge von Daten, die er neu anordnet, dann interpretiert und mit Hilfe geeigneter Abbildungen so ein einfaches Bild schafft. Nur am Rande sei bemerkt, dass man behaupten könnte, dies ähnele entfernt der allgemeinen Situation des menschlichen Umgangs mit dem Computer: Der Mensch erhält eine Menge von Daten vorgesetzt, die er nach kaum zu durchschauenden Richtlinien (also „zufällig“) ordnet und interpretiert, um sich daraus ein Bild zu machen.

Ein Lernlabor sollte erstellt werden. Bleibt die Frage, was man hier lernen kann.

Nachdem man bei der ersten Benutzung vermutlich verwirrt zurückbleiben wird, sollte man bei wiederholter Benutzung in der Lage sein, Zusammenhänge zwischen Bild und Befehlszeilen zu erkennen. Genauer: Die in der obenstehenden Tabelle abgebildeten Beziehungen zwischen den Bits und dem erzeugten Bild zu erkennen.

Hilfestellung erhält der Benutzende dabei durch die Zuordnung der Gruppen in den Befehlszeilen zu Aktionen auf der Bildfläche durch bewegte Bildlichkeit. Theorien des Benutzenden über die jeweiligen Zusammenhänge lassen sich durch wiederholtes Ausführen bestätigen oder widerlegen.

Als Lerneffekt denkbar sind Einblicke dar-

über, wie Daten (hier die eines Bildes) gespeichert werden, nämlich als Folgen von Nullen und Einsen. Denkbar sind ebenfalls Einblicke in die Art und Weise, wie aus diesen Daten Bilder erzeugt werden, nämlich durch die Ausführung eines Algorithmus, hier dargestellt durch die Abarbeitung der zwölf Befehlszeilen.

Nun wird der Benutzende wenig Nutzen davon haben, die simple Pseudo-Maschinensprache dieses Lernlabores zu verinnerlichen. Vielleicht ist es ihm aber möglich, die Daten auf das allgemeine Verhältnis zwischen Bild, Wahrnehmung und Ästhetik einerseits und rechnerinterner Abbildung, dem Algorithmischen, andererseits zu übertragen.

Dies könnte zu einem besseren Verständnis führen, wie in der algorithmischen Computerkunst Bilder erzeugt werden, bei besonderer Betonung des Elements des Zufalls, das in der Computerkunst eine tragende Rolle spielt.