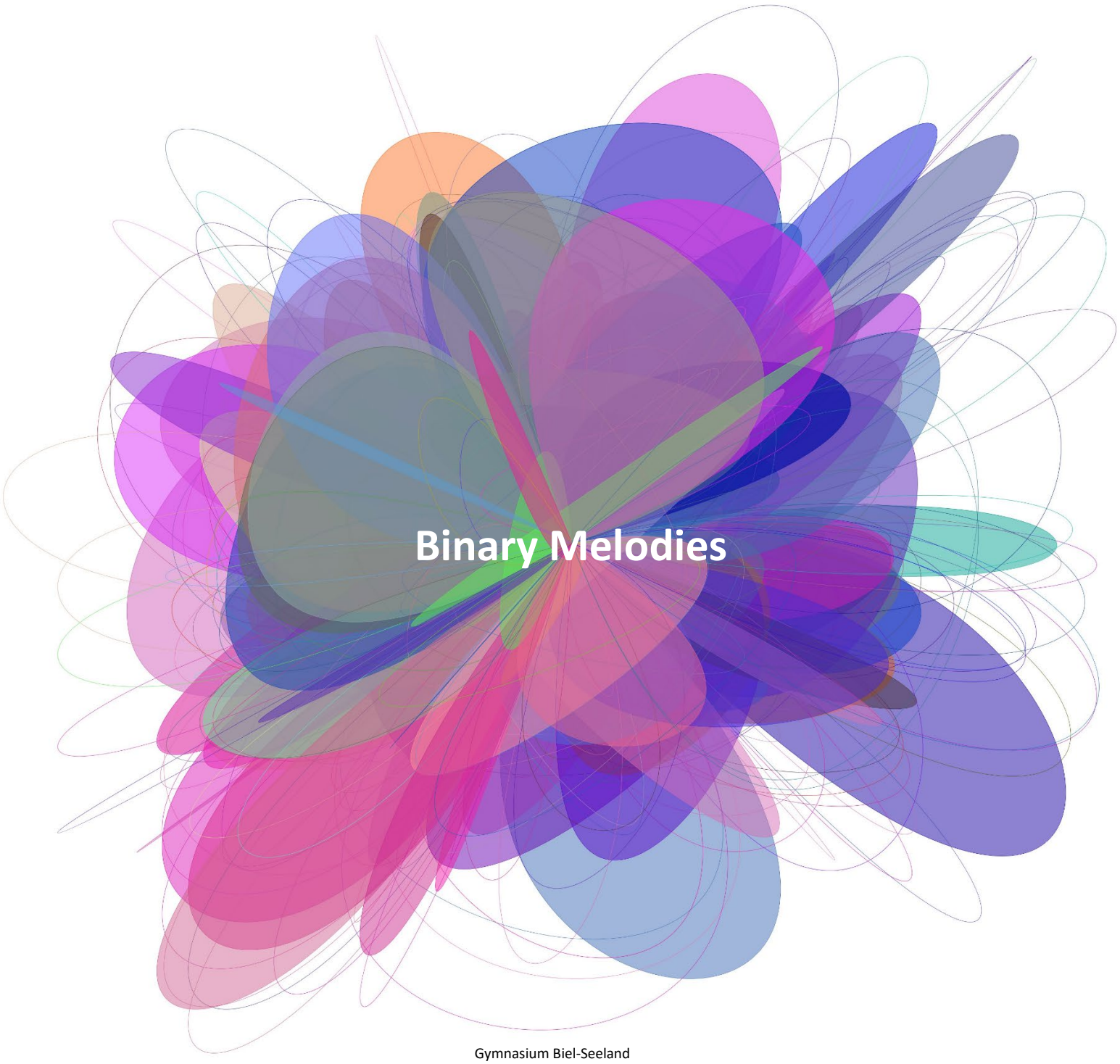


Binary Melodies

Gymnasium Biel-Seeland

Maturjahrgang 2024

Klasse: 24i

Autor: Alix Franck

Betreuer: Oliver Kreuter

Abstract

Diese Arbeit ist Teil eines umfassenden Kunstprojekts und fokussiert sich auf die Umsetzung des Zufallsprozesses als Methode zur Transformation eines Textes in ein visuelles Medium. Das Projekt setzt sich aus vier Komponenten zusammen:

1. Die Erkundung der Rolle der Zufälligkeit in der Digitalen Kunst, sowohl im kunsthistorischen Kontext als auch in Bezug auf zukünftige Anwendungen.

2. Die technische Herleitung von Bildern aus Texten, welches Schwerpunkt dieser Maturarbeit ist.

3. Die Website binarymelodies.com als zentrale Drehscheibe des Gesamtprojektes.

4. Eine Sammlung von NFTs (Non Fungible Tokens), die auf den generierten Bildern basieren, begleitet von Texten lokaler Künstlerinnen und Künstler aus Biel.

Der Verkaufserlös der NFTs kommt der Sozialen Organisation Passepartout Biel/Bienne zugute.

Im Zentrum steht die Frage: «Wie lässt sich der Zufallsprozess als Methode zur Transformation eines Textes in ein visuelles Medium implementieren?». Durch die Anwendung von Zufallsgeneratoren werden Schweizer Musiktexte in eine binäre Sprache überführt und anschliessend in visuelle Formen, insbesondere Bézier-Kurven, übersetzt. Eigens entwickelte Algorithmen und Programme demonstrieren eindrücklich, dass der Zufall ein Schlüsselelement für die Transformation von Text in einzigartige visuelle Kunstwerke darstellt.

Vorwort

Im Rahmen meines Schwerpunktfaches «Bildnerisches Gestalten» erhielt ich im Oktober 2022 den Auftrag, ein Werk mittels Zufalls zu erstellen. Ich programmierte zu diesem Zweck ein kurzes Java-Programm. Dieses verteilte Pixel auf einer 400x400mm grossen Leinwand mittels Zufalls-Funktion. Diese Pixel konnten beliebig eingefärbt werden, wiederum mittels Zufalls-Funktion. Diese Arbeit hat mich dazu inspiriert, meine Maturaarbeit im digitalen Bereich zu verfassen, in Kombination mit Zufallsoperationen in der Programmierung.

Besonderer Dank gilt meinem Betreuer, Oliver Kreuter, der mich während des gesamten Prozesses unterstützt und begleitet hat. Auch danke ich meiner Familie und meinen Freundinnen und Freunden für ihre Unterstützung und Ermutigung.

Mir war bewusst, was mein Unterfangen an technischem Wissen verlangte. Ich musste mich intensiv mit der Programmiersprache *Processing* befassen und habe mir dabei eine Grundlage erarbeitet, die für spätere Anwendung genutzt werden kann.

Abschliessend hoffe ich, dass diese Maturaarbeit als wertvolle Ressource für alle dienen kann, die sich für Zufall in der digitalen Kunst interessieren, und dass sie zu weiteren Diskussionen und Untersuchungen in diesem Bereich anregen wird. Ich freue mich darauf, die Ergebnisse und Erkenntnisse dieser Arbeit mit anderen zu teilen und bin offen für Feedback und konstruktive Kritik.

Biel, 24.10.2023

Alix Franck

Einleitung

Die vorliegende Arbeit Binary Melodies erforscht die Digitalisierung von Texten, insbesondere denen von Musikern aus der Stadt Biel.

Die Arbeit gliedert sich in verschiedene Teile, beginnend mit einer theoretischen Grundlage, welche die zu untersuchende Frage klar umreisst: «**Wie lässt sich der Zufallsprozess als Methode zur Transformation eines Textes in ein visuelles Medium implementieren?»**. Das Konzept des Zufalls in der digitalen Kunst aus mathematischer und historischer Perspektive beleuchtet wird an drei Künstlern veranschaulicht. Die Verwendung der Programmiersprache *Processing* stellt die Methode dar, deren Syntax wird zum Verständnis der praktischen Arbeit kurz erläutert. Diese wird durch eine dokumentarische Darstellung des Entwicklungsprozesses der fertigen Programme veranschaulicht. Diese Programme bilden die Basis für die Erstellung des Endproduktes. Der Output dieser Programme wird in dieser Arbeit beispielhaft vorgestellt.

Das Endprodukt wird auf der Webseite «binarymelodies.com» präsentiert. Durch die Kombination von theoretischen und praktischen Ansätzen bietet diese Arbeit eine umfassende Analyse der Möglichkeiten und Grenzen der graphischen Darstellung von Texten mittels Zufallstechniken.

Die Integration der *NFT-Technologie* (Non-Fungible Tokens) dient zur Distribution und Wertsteigerung der erzeugten Werke und zudem als Präsentationsmethode. Die Versteigerung dieser NFTs fördert nicht nur den ökonomischen Wert, sondern auch soziales Engagement, da die Einnahmen wohltätigen Zwecken zugutekommen. Alle Einnahmen des Projekts werden der Sozialen Institution *Passepartout- Biel* gespendet.

Zum besseren Verständnis der oben verwendeten Fachbegriffe Website, NFT, Blockchain und Processing wird im Anhang auf Online-Fachartikel Artikel verwiesen.

Grundlagen

In diesem Kapitel werden die theoretischen Grundlagen erläutert, die für das Verständnis der praktischen Anwendung erforderlich sind. Dazu gehören eine Definition des Zufalls in der Mathematik, Porträts von drei Künstlern, die bereits früh mit digitalen Zufallsmethoden arbeiten, sowie die grundlegende Syntax der Programmiersprache *Processing*.

Zufall

In der Mathematik bezeichnet der Begriff «Zufall» ein Phänomen, das nicht vorhersehbar oder berechenbar ist. Er dient oft zur Beschreibung von Ereignissen, die sich nicht exakt bestimmen lassen, wie beispielsweise Kursschwankungen an der Börse oder radioaktiver Zerfall (vgl. Knuth, 1997, S. 1). Es ist entscheidend, zwischen echter und mathematischer Zufälligkeit zu unterscheiden. Echte Zufälligkeit entsteht aus unberechenbaren physikalischen Prozessen. Im Gegensatz dazu erzeugen Algorithmen pseudozufällige Sequenzen, die statistisch kaum von echten Zufallssequenzen zu unterscheiden sind. Die Abwesenheit von Mustern ist ein Schlüsselmerkmal von Zufall. In einem vollkommen zufälligen Datensatz sollten keine erkennbaren Muster oder Wiederholungen auftreten. Diese Eigenschaft macht Zufallssequenzen besonders nützlich in Bereichen wie der Kryptographie (Verschlüsselungstechnik), wo Vorhersagbarkeit vermieden werden muss.

Die Bedeutung von Zufälligkeit erstreckt sich über zahlreiche Anwendungsgebiete. Dazu gehören maschinelles Lernen, künstliche Intelligenz und digitale Kunst.

Geschichte

In diesem Abschnitt wird die Thematik im kunsthistorischen Kontext dargelegt.

Eine weitläufige und zugleich tiefgreifende Erforschung der Wurzeln der digitalen Kunst führt uns zurück in die 1960er Jahre, die von den Fortschritten in der Computertechnologie geprägt war. In jenem kreativen Umfeld entwickelten sich die ersten Ansätze dessen, was wir heute als generative Kunst kennen - eine künstlerische Praxis, die sich aus der Verschmelzung von Technologie und Kreativität entwickelte. Während diese Periode auf die 1960er Jahre datiert ist, hat das Konzept der generativen Kunst seither eine rasante und facettenreiche Entwicklung erlebt (DAM, 2021). In den frühen Tagen der Computertechnologie eröffneten sich neue künstlerische Horizonte, die von einer kleinen Gruppe visionärer Mathematiker, Ingenieure und bildender Künstler erkundet wurden.

Laut (DAM, 2021) . . .

«. . . stammen die ersten algorithmischen Zeichnungen aus den 1960er und 1970er Jahren. Diese wurden von Personen geschaffen, die entweder in Forschungseinrichtungen mit Grossrechnern arbeiteten oder sich sogar in das aufregende Unterfangen stürzten, ihre eigenen Maschinen zu bauen.»

In der Folge dieser Pioniere haben viele Künstler den Einsatz von Algorithmen in ihren Arbeiten weiterentwickelt. Dabei hat insbesondere das Element des Zufalls eine zentrale Rolle gespielt.

In dieser Arbeit wird die Verwendung des Zufalls in der digitalen Kunst anhand der Arbeiten von drei Künstlern nachgezeichnet, die einen bedeutenden Beitrag zur Geschichte der Generativen Kunst geleistet haben.

Ein Hinwies zu Generativer Kunst findet sich im Anhang.

Mohr

Manfred Mohr, geboren 1938 in Deutschland, begann seine künstlerische Karriere in den 1960er Jahren und nutzte die damals neu aufkommenden Technologien, um eine ganz neue Form der Kunst zu schaffen - die algorithmische Kunst (Mohr, 2022).

Sein erstes Programm war ein Zufallszahlengenerator. Bemerkenswert ist, dass dieses Programm fehlerhaft war und nur die Zahl «1» generierte (Mohr, 2022) Dieser Fehler markierte den Beginn seiner Experimente in der Algorithmischen Kunst. Mohr selbst sagt dazu:

Abbildung 1: Output von Manfred Mohrs erstem

Laut Mohr (2022):

«*Mein Interesse daran, die Logik meiner Arbeit zu entdecken, führte dazu, dass ich eine sehr innige Beziehung zum Schreiben meiner Algorithmen hatte und mich selbst durch diesen Prozess entdeckte.»*

Mit der neu entdeckten Faszination für eigene Algorithmen entwickelte Mohr immer komplexere Programme, jedoch mit qualitativ stagnierendem Output. Erste Werke aus dem Jahr 1969 wurden mithilfe von Zufallszahlengeneratoren erstellt. Abbildung 2 zeigt «Random Number Collage 1», welche mithilfe von Computer und dem Künstler selbst erstellt wurden. Der Algorithmus zu diesem Bild funktioniert folgendermassen.

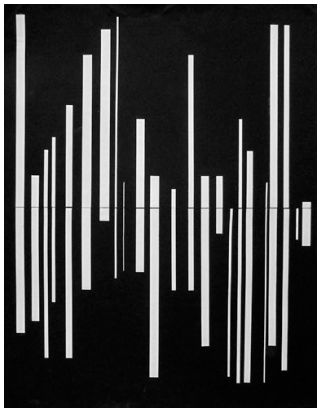


Abbildung 2 «Random Number Collage 1» Manfred Mohr, 1969

Laut Mohr (2022):

«Das Programm platziert weisse rechteckige Linien um eine zentrale Achse (x-Achse). Die Position, Höhe, Breite und sogar das Vorhandensein dieser Linien werden durch Zufallszahlen bestimmt.»

Mohr besass kein Gerät, welches Output produzieren konnte. So musste er die gegebenen Masse der Rechtecke von Hand auf die Leinwand bringen (Mohr, 2022).

Ein weiteres Werk aus dem Jahr 1969, in Abbildung 3, befasste sich ebenfalls mit Zufallszahlen. Auch bei diesem Bild verwendete er seine Hand als Output Methode.

Abbildung 3: «Random Walk», Manfred Mohr, 1969

Laut Mohr (2022):

«Der Algorithmus zeichnet einen Zufallsweg aus 100 abwechselnd horizontalen und vertikalen Linien in einem begrenzten Raum. Die horizontalen Linien sind in der Mehrzahl deutlich breiter. Die Linien ändern ihre Richtung, sobald sie an eine Begrenzung des Raums stossen.»

Diese Verwendung von Zufallsoperationen in der digitalen Kunst hatte grossen Einfluss. Im Jahr 1969 war das Erstellen eines Algorithmus eine schwierige Aufgabe, die umfangreiches Wissen im Bereich der Mathematik erforderte.

Die Werke Mohrs zählen zu den Pionierarbeiten der Computerkunst und fanden frühzeitig Eingang in Museumssammlungen. Sie sind mittlerweile weltweit in renommierten Sammlungen vertreten, unter anderem im *Centre Pompidou* in Paris, im *Victoria and Albert Museum* in London sowie im *Stedelijk Museum* in Amsterdam. Auch in weiteren Sammlungen von Tel Aviv über Berlin bis hin zu verschiedenen Standorten in den USA sind seine Arbeiten zu finden. (Spalter Digital, 2023)

Verostko

Der 1929 in den USA geborene Künstler Roman Verostko ist eine bedeutende Persönlichkeit in der Welt der algorithmischen Kunst. Verostko begann seine Karriere mit traditionellen Kunstformen und Medien, fand aber bald seine wahre Berufung in der digitalen Kunst (Verostko,2023).

Eines seiner ersten Ausstellungsstücke, «Die magische Hand des Zufalls», wurde auf einem 25-Zoll-Monitor präsentiert, der an einen Computer angeschlossen war. Was die Betrachter sahen, war eine flüssige, sich ständig verändernde Landschaft von Worten und Bildern, die alle in Echtzeit generiert wurden (Verostko,2023).

Laut Verostko (2007):

«Die Magische Hand des Zufalls war in der Lage, eine endlose Abfolge von originellen visuellen Themen zusammen mit originellen Titeln und gelegentlichen Sequenzen einer Art Computer-"Weisheits"-Literatur zu erzeugen, die ich als "Sprüche des Omphalos" bezeichnete. Diese Sequenzen waren in der Tat eine Form des Computerautomatismus, ein elementarer Schritt in Richtung künstliches Leben und künstliche Intelligenz»

Abbildung 2: Ein Frame von " The Magic Hand of Chance", Roman Verostko, 1982

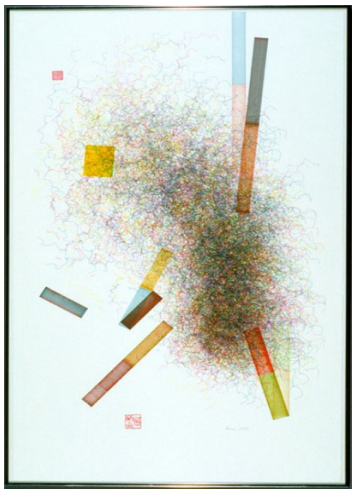


Abbildung 3. Fig 7, Hodos, Roman Verostko, 1988

Mit der Zeit begann Verostko, sich für Hard Copy zu interessieren und entwickelte Software, um bildnerische Elemente mit Linien und Pinselstrichen zu erstellen.

Dies führte zur Entwicklung von *Hodos*, einem Programm, das den Computer dazu brachte, Gemälde zu erstellen. *Hodos* ist ein integriertes Set von Algorithmen, mit Befehlen für einen Plotter . Verostko liess sich unter anderem von der chinesischen Kalligrafie inspirieren und strebte danach, eine spontane und expressive Energie in seinen Werken zu erzeugen. (Verostko,2023)

Barbadillo

Manuel Barbadillo, geboren 1929, war ein Pionier der digitalen Kunst in Spanien. Er erkannte früh das immense kreative Potenzial der Computertechnologie. Er gilt als der erste spanische Künstler, der die Möglichkeiten des Computers in seine Kunst integrierte.

Die Zusammenarbeit von Manuel Barbadillo mit dem *Centro de Cálculo* an der Universität Madrid im Jahr 1968 markierte eine bedeutende Phase in seiner Karriere, in der er die Integration von Computertechnologie mit künstlerischer Kreation erforschte (Galería Rafa, o.J.). Während dieser Zeit begann Barbadillo, Computer als künstlerische Instrumente zu verwenden, was ihm ermöglichte, in eine neue Sphäre künstlerischer Exploration einzutauchen, die durch eine Mischung aus mathematischen und computergestützten Konzepten mit traditionellen künstlerischen Elementen gekennzeichnet war (AtariArchives.org, o.J.).

Ein entscheidender Aspekt dieser Zusammenarbeit war die Entwicklung einer Bibliothek von Grundformen oder Modulen. Diese Bibliothek, besser ausgedrückt eine Sammlung mathematischer Funktionen zur Definition von Formen, nutzte Barbadillo, um rhythmische Muster in seiner Kunst zu schaffen. Sie kann als digitales Repositorium angesehen werden, das bei der Erzeugung komplexer und ästhetisch ansprechender Muster half (Spalter Digital Art, o.J.).

Abbildung 4: Adfera, Manuel Barbadillo, 1972

Das Kunstwerk *Adfera*, in Abbildung 4, ist ein gutes Beispiel für seine explorative Arbeit mit Computern. *Adfera* ist ein Druck, der durch die Verwendung von Computeralgorithmen erstellt wurde, welche in der Programmiersprache FORTRAN – eine der ältesten Programmiersprachen – geschrieben wurden (Spalter Digital Art, o.J.). Diese Algorithmen wurden im Computerzentrum der Universidad de Madrid ausgeführt, wobei Barbadillo die Ergebnisse dann als Basis für seine Drucke nutzte. Die spezifischen Algorithmen, die Barbadillo für *Adfera* verwendet hat, wurden entwickelt, um eine Vielzahl von geometrischen Formen und Mustern zu erzeugen. Durch die Manipulation von Variablen innerhalb des Codes konnte Barbadillo eine breite Palette von visuellen Effekten und Kompositionen erzielen (Spalter Digital Art, o.J.).

Trotz seiner Vorliebe für Computertechnologie schätzte Barbadillo den Wert handgefertigter Kunst und zog es vor, die finalen Versionen seiner Werke von Hand zu vervollständigen. Dies schuf einen faszinierenden Kontrast zwischen moderner Technologie und traditioneller Handwerkskunst, wie im Beispiel von Manfred Mohr. Sein Erbe inspiriert weiterhin Künstlergenerationen, die Technologie und Tradition verbinden möchten (Spalter Digital Art, o.J.).

Barbadillos systematischer und strukturierter Ansatz zur Erkundung von Modulpermutationen und -symmetrien bietet eine interessante Perspektive in der Untersuchung der Rolle des Zufalls in der digitalen Kunst. Seine Arbeit stellt einen kontrollierten Kontrast zu zufälligeren Ansätzen dar und könnte als Vergleichsgrundlage dienen, um zu verstehen, wie Zufall und Kontrolle die ästhetischen und strukturellen Ergebnisse in der digitalen Kunst beeinflussen. Somit setzte Pionier Barbadillo im Gegensatz zu Mohr und Verostko den Zufall nicht im gleichen Mass als Gestaltungselement ein.

Fazit

Die digitale Kunst, wie sie von Pionieren wie Manfred Mohr, Roman Verostko und Manuel Barbadillo praktiziert wurde, hat die Grenzen dessen, was in der Kunst als möglich und akzeptabel gilt, neu definiert. Diese Künstlerinnen und Künstler waren nicht nur Zeugen einer technologischen Revolution, sondern auch aktive Mitgestalter, die frühzeitig die neuen Möglichkeiten erkannten und in ihre künstlerischen Praktiken integrierten. Ihre Werke sind nicht nur ästhetisch beeindruckend, sondern auch intellektuell herausfordernd, da sie traditionelle Vorstellungen von Kunst, Künstlerschaft und dem kreativen Prozess in Frage stellen.

Ein zentrales Element in ihren Werken ist der Einsatz von Zufall. Manfred Mohr verwendete Algorithmen, um Kunstwerke zu schaffen, die sowohl zufällig als auch präzise sind. Roman Verostko betrachtete den Zufall als eine Art magische Hand, die in der Lage ist, endlose Reihen von originellen visuellen Themen zu generieren. Manuel Barbadillo kombinierte Zufall mit präzisen Algorithmen, indem er Permutationen auf seine Formenbibliothek anwandte.

Der Zufall, wie er von diesen Künstlerinnen und Künstlern verwendet wurde, hinterfragt traditionelle Konzepte von Kontrolle und Absicht in der Kunst. Er ermöglicht es ihnen, Werke zu erschaffen, die sowohl organisch als auch unvorhersehbar sind. Dadurch entsteht eine Kunstform, die in einer «gemeinsamen Anstrengung» sowohl vom Künstler als auch von der Maschine geschaffen wird.

Was diese Künstlerinnen und Künstler im Kontext dieser Arbeit kunsthistorisch relevant macht, ist nicht nur ihre Fähigkeit, Technologie und Zufall in ihre Kunst zu integrieren. Sie präsentieren auch Visionen davon, wie Kunst in einer zunehmend digitalen Welt aussehen könnte. Sie haben nicht nur neue Techniken und Medien in das Kunstschaffen eingebracht, sondern auch neue Fragen zur Rolle des Künstlers, zur Natur der Kunst und zur Beziehung zwischen Menschen, Maschinen und Zufall gestellt. Ihre Werke fungieren sowohl als Reflexionen ihrer Zeit als auch als Vorhersagen für die Zukunft. Sie ebnen den Weg für nachfolgende Generationen von Künstlerinnen und Künstlern, die in einer Welt aufwachsen, in der die Grenzen zwischen Kunst und Technologie immer mehr verschwimmen.

Zukunft

Die generative Kunst hat sich dank der Integration von Algorithmen und Zufallsfaktoren als spannendes Feld etabliert. Mit fortschrittlichen Technologien, insbesondere Künstlicher Intelligenz (KI) und Maschinellem Lernen, wurden die Möglichkeiten dieser Kunstform erweitert. Nun können KI-basierte Algorithmen Bilder erzeugen, die kaum von menschlich geschaffenen Kunstwerken zu unterscheiden sind. Diese Entwicklungen könnten die Rolle des Zufalls in der Kunst weiter vertiefen und die Frage der Authentizität neu definieren (Cryptopolitan, 2021).

In diesem Kontext gewinnt die Frage der Authentizität und des Eigentums an Bedeutung, da digitale Kunstwerke und ihre Schöpferinnen und Schöpfer immer zahlreicher werden. Non Fungible Tokens (NFTs) repräsentieren einen wesentlichen Wandel, da sie die Blockchain-Technologie nutzen, um jedem digitalen Kunstwerk eine unveränderliche, eindeutige Signatur zu verleihen. Dies sichert die Authentizität und ermöglicht eine neue Art der Interaktion zwischen Künstlerinnen und Künstlern, Sammlerinnen und Sammlern sowie dem Kunstwerk selbst (Cryptopolitan, 2021).

Das NFT-Projekt *Art Blocks* ist eine innovative Plattform, die einen Raum für generative Kunst durch die Verbindung von Blockchain-Technologie schafft. Auf dieser Plattform können Künstlerinnen und Künstler ihre eigenen Algorithmen erstellen, die als Grundlage für Kunstwerke dienen. Wenn Sammlerinnen und Sammler ein Kunstwerk erwerben möchten, wird durch einen Prozess namens «Prägen» (mint) ein einzigartiges Kunstwerk geschaffen. Dieses Kunstwerk wird dann auf der Ethereum-Blockchain gespeichert und kann von der Künstlerin oder dem Künstler, den Sammlerinnen oder Sammlern und der gesamten Gemeinschaft gleichzeitig betrachtet werden (Art Blocks, o.j.).

Art Blocks hostet ausgewählte Sammlungen von generativer NFT-Kunst, die zum Verkauf stehen. Wenn ein Kunstwerk verkauft wird, wird es als Non Fungible Token (NFT) geliefert, was den Käuferinnen und Käufern eine verifizierbare Eigentümerschaft am Kunstwerk ermöglicht (DappRadar, o.j.). Beim «Prägen» eines Kunstwerks auf *Art Blocks* wird tatsächlich ein einzigartiges Token auf der Ethereum-Blockchain erstellt. (101 Blockchains, o.j.)

Ein Hinwies zu obigen Fachausdrücken (Blockchain, prägen/Minting, Non fugitable Token/ NFT, Künstliche Intelligenz/KI) findet sich im Anhang.

Processing

In diesem Kapitel wird die grundlegende Funktionsweise der Programmiersprache *Processing* erläutert. Ein herausragendes Beispiel für den Einsatz von Algorithmen in der Kunst ist eben diese Programmiersprache. *Processing* ist eine Open-Source-Programmiersprache, die speziell für Künstlerinnen, Künstler und Designer entwickelt wurde. Sie ermöglicht die Erstellung von interaktiven softwarebasierten Kunstwerken (vgl. Reas & Fry, 2014, S. 1). Die Möglichkeit, Zufallsgeneratoren und Algorithmen zu integrieren, fügt dem kreativen Prozess eine zusätzliche Ebene der Komplexität hinzu.

Syntax

Die Syntax von *Processing* stellt eine Sammlung von Regeln dar, die die Struktur eines Programms festlegen (vgl. Reas & Fry, 2014, S. 12). In den folgenden Abschnitten wird lediglich der für diesen Kontext relevante Syntax behandelt, insbesondere Datentypen und Funktionen. Für ein tiefergehendes Verständnis kann die offizielle Processing-Dokumentation aufgerufen werden - Verlinkung im Anhang.

Variablen

Variablen dienen zur Speicherung verschiedener Datentypen. Insbesondere spielen die Datentypen `string`, `float` und `int` eine zentrale Rolle. `String` wird für die Speicherung und Manipulation von Textzeichenfolgen verwendet. Der Datentyp `float` kommt bei Gleitkommazahlen zum Einsatz und ermöglicht eine präzise Darstellung von Dezimalzahlen. Das ist insbesondere wichtig für die Wiedergabe von kontinuierlichen Werten, wie sie in der digitalen Kunst für Farbverläufe, Bewegungen und andere dynamische Elemente benötigt werden. `int` wird für die Speicherung von ganzen Zahlen verwendet (vgl. Greenberg, 2007, S. 63-67). Diese Datentypen finden Anwendung im späteren Kapitel, wo sie zur Datenspeicherung genutzt werden. Daher ist es notwendig, sie hier zu erläutern.

```
String = "Mars"; // PLanet

Float  = 3.71; // Gravitation

int  = 2; // Anzahl Monde
```

Funktionen

Funktionen sind Blöcke von Code, die eine bestimmte Aufgabe ausführen. In *Processing* sind `setup()` und `draw()` zwei grundlegende Funktionen (vgl. Reas & Fry, 2014: S 12). In der Entwicklung von Programmen mit *Processing* bilden die Funktionen setup() und draw() das Rückgrat des Programmflusses und sind entscheidend für die flüssige Ausführung von Code. Die `setup()`-Funktion wird einmal zu Beginn des Programms aufgerufen und dient der Initialisierung von Variablen, der Bildschirmgrösse und anderen Einstellungen. Die `draw()`-Funktion wird wiederholt aufgerufen und ist der Hauptteil des Programms, in dem die Grafiken gezeichnet werden.

```
void setup() {
  // Initialisierungscode
}
void draw() {
  // Hauptzeichencode
}
```

random()

Die `random()` Funktion in *Processing* ist ein wesentliches Werkzeug für die Erzeugung von Zufallszahlen. Sie ermöglicht die Generierung von Gleitkommazahlen innerhalb eines definierten Bereichs, was in der digitalen Kunst vielfältige Anwendungen findet. Die Funktion kann beispielsweise zur zufälligen Platzierung von Objekten, zur Farbauswahl oder zur Erzeugung von Koordinaten verwendet werden (vgl. Reas und Fry, 2014: S 127).

Die Syntax der `random()` Funktion ist einfach und flexibel. Sie kann mit einem oder zwei Argumenten aufgerufen werden, um den Wertebereich der erzeugten Zufallszahlen zu definieren (vgl. Reas und Fry, 2014: S 128).

```
random (100);  // Zufallszahl zwischen 0 und 100
random (50, 100);  // Zufallszahl zwischen 50 und 100
```

Es ist wichtig zu beachten, dass die `random()` Funktion in *Processing* pseudozufällige Zahlen erzeugt. Diese werden im folgenden Abschnitt erläutert.

randomSeed()

Computer sind Maschinen, die beständige und genaue Berechnungen durchführen und daher Zufallszahlen simulieren müssen, um die Art von Zufall zu simulieren (vgl. Reas & Fry, 2014: S 129-130). Das Konzept des «RandomSeed» ist bei Erzeugung von pseudozufälligen Zahlen essenziell. Ein Seed (Saat) ist ein Anfangswert, der als Eingabe für einen Pseudozufallszahlengenerator PRNG (pseudorandom number generator). PRNGs sind Algorithmen, die deterministisch arbeiten und eine Sequenz von Zahlen erzeugen, die zufällig erscheinen, es jedoch nicht sind. Der Seed-Wert dient als Ausgangspunkt für diese Sequenz, und die resultierenden Zahlen sind vollständig von diesem Anfangswert abhängig (Processing 3.0 Documentation, o.j.).

Ein einfaches Beispiel für einen PRNG ist der Lineare Kongruenzgenerator (LCG). Die Formel für einen LCG ist:

$$X\{n+1\} = (a \times Xn + c) \ mod\, m$$

$X\{n+1\}$ für die nächste Zahl in der Sequenz,

Xn für die aktuelle Zahl (oder den Seed, wenn es die erste Zahl ist),

a, c und m sind Konstanten, die den Generator charakterisieren

(vgl. Knuth, 1969: S 10)

Ein einfaches Beispiel kann die Funktionsweise eines linearen Kongruenzgenerators (LCG) verdeutlichen. Nehmen wir an, die Konstanten a, c und m sind 5, 3 und 16, und der anfängliche Saat-Wert X_0 ist 1.

Der Algorithmus berechnet dann den nächsten Wert X_{n+1} in der Sequenz wie folgt:

$$\mathrm{X1} = (5 \times 1 + 3) \bmod 16 = 8$$

$$\mathrm{X2} = (5 \times 8 + 3) \bmod 16 = 11$$

$$\mathrm{X3} = (5 \times 11 + 3) \bmod 16 = 14$$

$$X4 \rightarrow Xn$$

Die Sequenz der Zahlen ist fest definiert und wird sich wiederholen, sobald ein Zustand erreicht wird, der schon einmal aufgetreten ist. Man kann sehen, dass durch die Einstellung eines anfänglichen Seed-Wertes die resultierende Sequenz vollständig festgelegt wird. In diesem Beispiel würde die Einstellung des Seeds auf 1 immer die gleiche Sequenz von Zahlen erzeugen: 1, 8, 11, 14 bis Xn.

Die tatsächlichen Zahlen und Formeln in realen Anwendungen sind in der Regel viel komplexer, aber das grundlegende Prinzip bleibt gleich. In *Processing* wird die Funktion `randomSeed()` verwendet, um den Seed-Wert für den PRNG festzulegen. Sobald dieser festgelegt ist, wird jede nachfolgende Ausführung der `random()`-Funktion eine vorhersagbare und reproduzierbare Sequenz von Zahlen erzeugen, die auf dem Seed basiert. (vgl. Reas und Fry, 2014: S 130).

bezier()

Processing verfügt über sieben Funktionen, die bei der Erstellung einfacher Formen helfen. Im Rahmen dieser Arbeit liegt der Fokus auf der 'bezier()' Funktion (vgl. Reas und Fry, 2014: S 30).

Bézier-Kurven sind parametrische Kurven, die häufig in Computergrafik und Design verwendet werden. In *Processing* können Bézier-Kurven mit der Funktion 'bezier()' gezeichnet werden. Eine Bézierkurve wird in diesem Kontext durch vier Punkte definiert: zwei Endpunkte (x1, y1 und x2, y2) und zwei Kontrollpunkte (Cx1, Cx2 und Cy1, Cy2), welche die Krümmung der Kurve beeinflussen.

```
bezier(x1, y1, cx1, cy1, cx2, cy2, x1, y1);
```

Hier ein Beispiel für die Verwendung von Bézier-Kurven in Processing:

Code:

```
void setup() {
  // Leinwandgrösse einrichten
  size(400, 400);
  background(255); // Weisser Hintergrund

  // Rand- und Füllfarben definieren
  stroke(0); // Schwarze Umrandung
  fill(222); // Halbtransparente graue Füllung

  // Eine Bezier-Kurve zeichnen
  bezier(30, 200, 80, 100, 320, 300, 370, 200);
}
```

Output:

In diesem Beispiel beginnt die Bezier-Kurve am Punkt (30, 200) und endet am Punkt (370, 200). Die Kontrollpunkte sind (80, 100) und (320, 300).

Anwendung

In diesem Kapitel wird die praktische Arbeit durch eine dokumentarische Darstellung des Entwicklungsprozesses der fertigen Programme veranschaulicht. Diese bilden die Basis für die Erstellung des Endproduktes.

Der konzeptuelle Ansatz des Endproduktes ist: Zehn Liedtexte in Binärcode zu übersetzen und diesen als Grundlage zu verwenden, um grafische Medien zu zeichnen.

Programme

In diesem Kapitel werden die erstellten Programme genauer erläutert. Alle Programme haben die gleiche grundlegende Funktionsweise: Binär-Code in graphische Medien zu übersetzen. Der Output dieser Programme wird in dieser Arbeit nur beispielhaft vorgestellt. Das Endprodukt wurde mit den folgenden Programmen erstellt und ist auf der Webseite «binarymelodies.com» zu finden.

setup()

Alle Programme, die im Rahmen dieser Arbeit erstellt wurden, haben dieselbe ‘setup()’ Funktion. Diese ist verantwortlich für alle Voreinstellungen, Grösse der Zeichen Fläche, Daten Verarbeitung und Zufallszahlengenerator-Initialisierung.

```
setup() {
size(5000, 5000);
String[] lines = loadStrings("binaryData.txt");
String binaryString = join(lines, "");
float[] binaryData = new float[binaryString.length()];
for (int i = 0; i < binaryData.length; i++) {
binaryData[i] = binaryString.charAt(i) == '0' ? 0 : 1;
long seed = binaryString.hashCode();
randomSeed(seed);
}
```

Diese funktioniert folgendermassen:

Die `setup`-Funktion wird einmalig aufgerufen, um Initialisierungen vorzunehmen. Zuerst wird eine Zeichenfläche der Grösse 5000 x 5000 Pixel erzeugt. Dann wird eine Textdatei geladen, deren Inhalt zeilenweise in einem String-Array gespeichert wird. Anschliessend werden alle Inhalte der Anordnung zu einem einzigen String zusammengefügt, wobei sämtliche Zeilenumbrüche entfernt werden. Arrays in der Programmierung sind wie eine Perlenkette, bei der jede Perle einen bestimmten Wert repräsentiert und durch ihre Position in der Kette identifiziert wird.

Beispiel: Input aus der Textdatei: String = ‘01010101 01101101’ => ‘0101010101101101’

Im nächsten Schritt wird ein neuer Float-Array namens `binaryData` erstellt, welcher die gleiche Länge hat wie der neu gebildete String. Am Ende erhält man einen Float-Array, der die binären Daten aus der Textdatei repräsentiert. Jede 0 oder 1 aus der Textdatei wird dabei in die Float-Anordnung übernommen.

Der Übergang von 'String' zu 'Float' ist in diesen Situationen notwendig. Ein String speichert Zeichen, einschliesslich Zahlen, aber als Text. Um mathematische Operationen mit diesen Zahlen durchzuführen, müssen sie in einen numerischen Datentyp wie 'Float' (vgl. Kapitel Variablen) umgewandelt werden. In dem beschriebenen Szenario wird durch Überprüfung jedes Zeichens im String und Zuordnung einer 0 oder 1 in der Float-Anordnung eine binäre Darstellung erzeugt.

In der `setup()` Funktion wird der Zufallszahlengenerator mittels zwei entscheidender Schritte initialisiert, um bei jedem Programmstart mit dem gleichen binären Input stets denselben Output zu erzielen. Hierbei wird die `randomSeed()` Funktion genutzt. Im Folgenden werden die zwei Schritte erklärt:

```
 // Generiere einen Seed basierend auf den binären Daten
long seed = binaryString.hashCode();
randomSeed(seed); // Initialisiere den Zufallsgenerator mit dem Seed
```

1. Hash-Code Erzeugung:

   Im ersten Schritt `long seed = binaryString.hashCode();` wird der `hashCode()` Befehl auf die Variable `binaryString` angewendet, welche eine binäre Zeichenfolge beinhaltet. Durch die `hashCode()` Methode wird ein einzigartiger Wert, basierend auf dem Inhalt der `binaryString` Variable, erzeugt. Der resultierende Hash-Code wird in der Variable `seed` gespeichert. (Processing 3.0 Documentation, o.j).

   Zum Beispiel generiert der Hash-Code der binären Zeichenfolge ‘101010’ die Zahl ‘1448664861’. Diese Zahl kann nun als 'seed' verwendet werden.

2. Zufallszahlengenerator Initialisierung:

   Im zweiten Schritt wird die `randomSeed()` Funktion mit dem zuvor berechneten Hash-Code `seed` als Argument aufgerufen. Durch die Übergabe dieses Seed-Werts wird die Zufallszahlensequenz der `random()` Funktion festgelegt. Mit dem gleichen Seed-Wert wird bei jedem Programmstart die gleiche Zufallszahlensequenz erzeugt (vgl. randomseed).

Wichtig ist das dieser Codeabschnitt nur für die Füllung der Kurven verantwortlich ist und nicht für die Zeichnung der Kurven selbst. Die genauere Verwendung wird in Abschnitt ‘chance()’ erklärt. Zusammenfassend sorgt dieser Codeabschnitt für die Initialisierung des Zufallszahlengenerators in *Processing*. Der berechnete 'seed' erlaubt eine reproduzierbare Zufallsgenerierung, die an den gegebenen Binärcode gekoppelt ist.

Ein Hinwies zu Hash-Funktion findet sich im Anhang.

Auf Basis dieser `setup()` Funktion wurden verschiedene Programme entwickelt, welche die Binär-Code und Zufallszahlen nutzen, um grafische Elemente wie Kurven und Farben mathematisch anzuordnen. Der genaue Ablauf und weiter verwendete Funktionen werden an einem Beispiel am folgenden Programm ‘Graustufe’ erklärt.

Graustufe

Um Bézier-Kurven zu generieren, die vom binären Input abhängig sind, verwendet das Programm eine spezielle Struktur. Hier ist eine zusammengefasste und klare Erklärung des Prozesses:

draw()

Nach Initiierung der '**setup()**' Funktion wird die `**draw()**` Funktion aufgerufen.

Diese Funktion ist das Herzstück des Programms. Sie zeichnet Bézier-Kurven basierend auf den Binärdaten, die aus der Datei 'binaryData.txt' geladen wurden. Die 'Draw' Funktion teilt Die Binärdaten in 40-Bit-Pakete, für jedes dieser Pakete werden spezifische Berechnungen durchgeführt. Ein Paket wird folgendermassen berechnet:

1. Die ersten und zweiten 8-Bits werden in Dezimalzahlen umgewandelt und als Koordinaten cx1 und cy1 verwendet.

2. Die nächsten 8 Bits werden in eine Dezimalzahl umgewandelt und als Graustufenwert verwendet, der im Spektrum von 0 bis 230 liegt.

background (0);

background (124);

background (230);

3. Die letzten zwei 8-Bit werden in zwei Dezimalzahlen umgewandelt und als Koordinaten cx2 und cy2 für den Kontrollpunkt der Bézier-Kurve verwendet.

In jedem Durchlauf wird eine Bézier-Kurve gezeichnet, wobei die Farbfüllung auf dem Graustufenwert und einer Transparenz von 127 basiert. Die Kurve startet und endet in der Mitte des Fensters, wobei cx1, cy1 die ersten Punkte und cx2, cy2 die Kontrollpunkte für die Krümmung der Kurve sind.

```
bezier(width/2, height/2, cx1, cy1, cx2, cy2, width/2, height/2);
```

So werden nur Krümmung und Farbwert vom Binär Code beeinflusst. Start- und Endpunkt sind hier von der Berechnung ausgeschlossen, da Höhe und Breite durch zwei die Mitte der Leinwand ergeben.

binaryToDecimal()

Die `binaryToDecimal` Funktion ist ein zentrales Werkzeug für die Umwandlung von Binär- in Dezimalzahlen. Diese Umwandlung ist notwendig, um Koordinaten und Graustufenwerte der Bézier-Kurven zu bestimmen.

```
int binaryToDecimal(float[] binaryData, int startIndex, int length) {
  int decimal = 0;
  for (int i = 0; i < length; i++) {
    decimal += binaryData[startIndex + i] * pow(2, length - i - 1);
  }
  return decimal;
}
```

Hier eine vereinfachte Erklärung des Prozesses:

1. Funktionsprozess:
   - Die Funktion geht jedes Bit der Binärzahl durch.
   - Jedes Bit, entweder 0 oder 1, wird mit 2 hoch (Gesamtlänge der Bits - Position des Bits - 1) multipliziert.
   - Der errechnete Wert wird zur Gesamtsumme `decimal` hinzugefügt.
   - Nach Durchlauf aller Bits wird der Gesamtwert `decimal` zurück an die `draw()` Funktion geliefert, die dann die Kurve zeichnet.(GeeksforGeeks, o.j.)

Angenommen, wir haben die Binärzahl `1010`, die wir in eine Dezimalzahl umwandeln möchten.

Umrechnung:

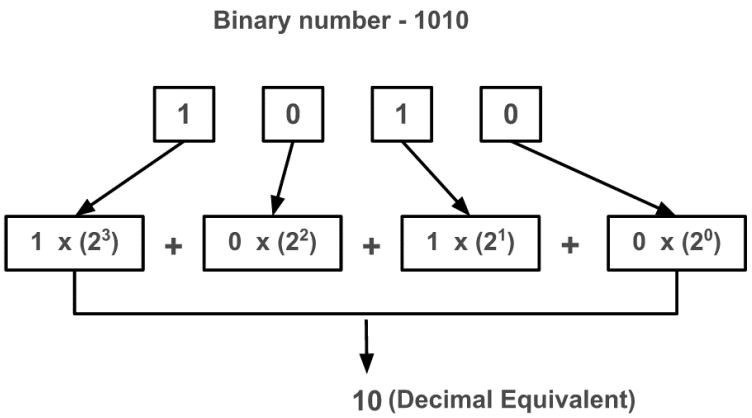


Abbildung 5: Binary to Decimal Diagramm, GeeksforGeeks

So wird die Binärzahl `1010` in die Dezimalzahl `10` umgewandelt. Die oben erklärte Funktion 'binaryToDecimal' ist in allen Programmen gleich.

In Kürze kann das Programm Graustufe so beschrieben werden:

Die binären Daten werden zuerst aus einer Textdatei geladen, in einen Float-Array umgewandelt und dann in dezimale Werte konvertiert, die als Koordinaten und Graustufenwerte für die grafische Darstellung verwendet werden. In der `draw()`-Funktion werden diese Werte genutzt, um Bezier-Kurven zu zeichnen, wobei die Krümmung und Farben der Kurven von den binären Daten abgeleitet werden, um schliesslich ein Bild zu erzeugen und zu speichern.

In einem Beispiel wurde der Text dieses Kapitels in Binärcode übersetzt und in das Programm ‘Graustufe’ gegeben.

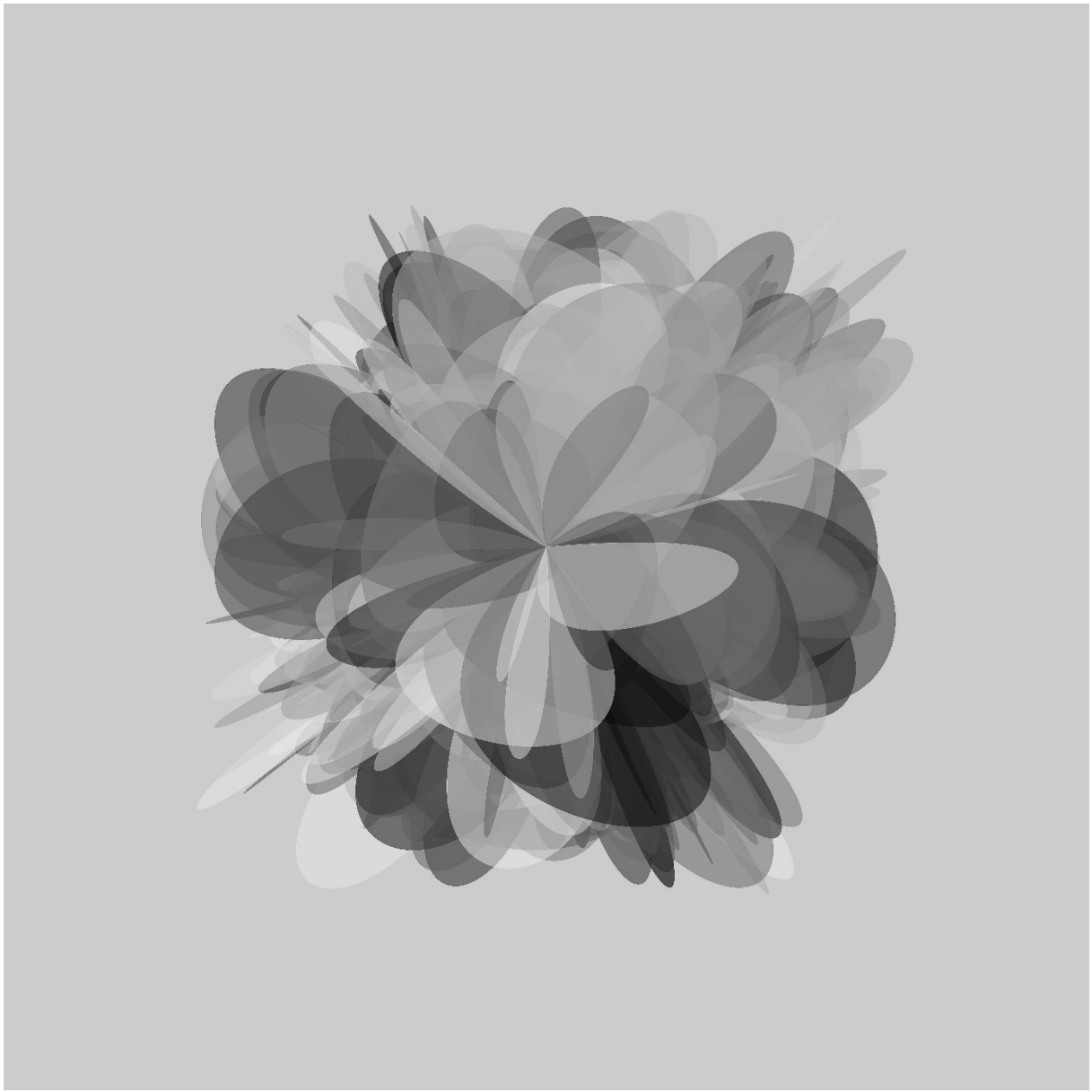


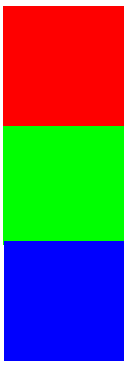
Abbildung 6 Output1 von ‘Graustufe’

RGB

Das Einbringen von Farbe ist möglich, indem man die `**draw()**` Funktion verändert und die Daten in grössere Zahlenpakete unterteilt werden. In diesem Programm werden Binärdaten aus einer Datei gelesen, in 56-Bit-Pakete unterteilt und zur Erzeugung von Bézier-Kurven verwendet, wobei nun auch RGB-Farbwerte berücksichtigt werden. Die Schritte sind wie folgt:

1. Die `draw` Funktion nimmt nun 56-Bit-Pakete anstelle von 40-Bit-Paketen. Die ersten beiden 8-Bit-Blöcke werden für die Koordinaten cx1 und cy1 verwendet.

2. Die nächsten drei 8-Bit-Blöcke werden in Dezimalzahlen umgewandelt, die als RGB-Werte (Rot, Grün, Blau) für die Farbe der Bézier-Kurven verwendet werden.



background(255, 0, 0);

background(0, 255, 0);

background(0, 0, 255);

3. Die letzten 8 Bits werden in zwei Dezimalzahlen umgewandelt, die als Koordinaten cx2 und cy2 für den Kontrollpunkt der Bézier-Kurve dienen.

Die `binaryToDecimal` Funktion bleibt für die Binär-zu-Dezimal-Umwandlung zuständig. In jedem Durchlauf zeichnet das Programm eine Bézier-Kurve mit einer Füllfarbe basierend auf den RGB-Werten und einer Transparenz von 127. Die Kurve beginnt und endet in der Fenstermitte, wobei cx1, cy1 und cx2, cy2 die Kontrollpunkte für die Krümmung sind. Das resultierende Bild wird als 'output2.jpg' gespeichert. Im Vergleich zum vorherigen 'Graustufe'-Programm ändert sich nur die `draw()` Funktion, der restliche Code bleibt unverändert.

In Kürze kann das Programm RGB so beschrieben werden:

Das Programm liest Binärdaten aus einer Datei, teilt diese Daten in 56-Bit-Pakete auf und verwendet diese, um Bézier-Kurven zu zeichnen. Die Umwandlung von Binär in Dezimal erfolgt durch die `binarytoDecimal` Funktion, und diese Dezimalzahlen werden dann als Koordinaten und Farbwerte in den Bézier-Kurven verwendet. Das resultierende Bild, im Format 5000x5000 Pixel, wird als ‘output2.jpg’ gespeichert. Im Vergleich zum ‘Graustufe’ Programm wird nur die `draw()` Funktion verändert, der Rest des Programms bleibt gleich. Der ganze Code ist im Anhang zu finden.

In einem Beispiel wurde der Text dieses Abschnitts in Binärcode übersetzt und in das Programm ‘RGB’ gegeben.

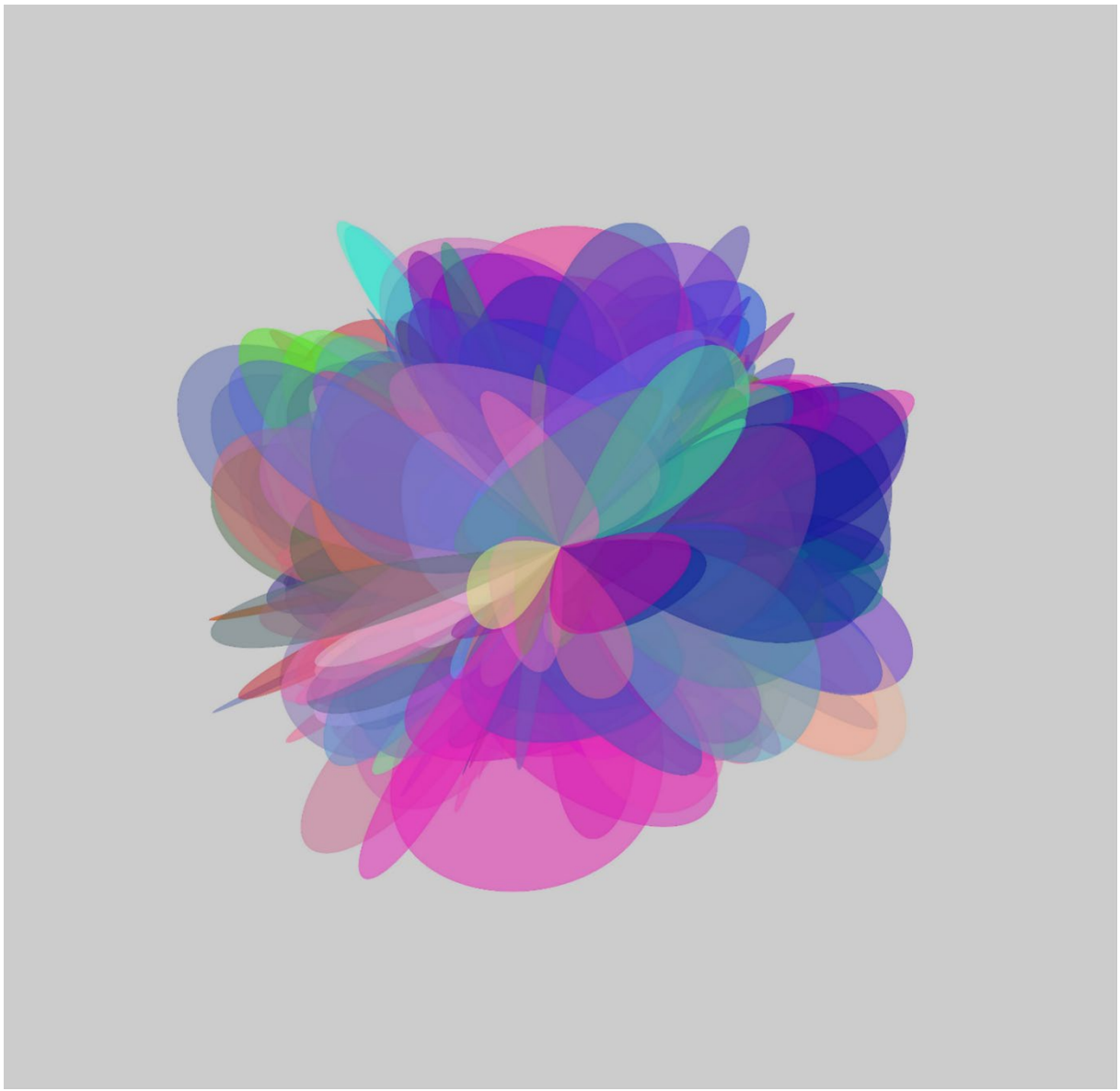


Abbildung 7. Output 2 von ‘RGB’

chance()

In den Outputs eins und zwei wird der Wahrscheinlichkeitsfaktor für den Füllwert der Kurven nicht betrachtet, so entstehen bis zu 800 gefüllte Kurven simultan. Dies lässt den Output vollständig wirken und kann den Eindruck verleihen, dass keine Zufalls Methode verwendet wurde.

Fügt man dieses Stück Code hinzu, wird nur ein Drittel der Kurven gefüllt:

```
float chance = random(1);
if (chance < 0.3) {
  fill(grayScaleValue, 127);
} else {
  noFill();
}
```

Der ‘randomSeed’ (vgl. setup) sorgt dafür, dass die Füllung mit den Binär-Daten übereinstimmt, wenn ‘chance = random(1)’ verwendet wird. Somit sind alle Kurven und Füllungen vom binären Input abhängig. Erhöht oder verringert man die Dezimalzahl der Variable ‘if (chance < 0.3)’ wird Einfluss auf den Zufall genommen.

Die Entscheidung, einen Output zu generieren, der nur teilweise gefüllt ist, bietet mehrere strategische Vorteile im Kontext dieser Arbeit. Erstens erzeugt die partielle Füllung der Form eine zusätzliche Ebene der Komplexität, die den Output weniger deterministisch macht. Dies ist besonders wertvoll für Anwendungen, die eine gewisse Unvorhersehbarkeit oder Einzigartigkeit in den generierten Mustern erfordern.

Zweitens bietet die partielle Füllung der Form eine ästhetische Qualität, die als Charakter bezeichnet werden könnte. Diese Unvollkommenheit kann als organische Form betrachtet werden, was das Endprodukt visuell interessanter und ansprechender gestaltet.

Drittens fördert die partielle Füllung die Diversität der Generierung. Anstatt homogener und wiederholbarer Muster bietet diese Methode die Möglichkeit, eine breitere Palette von Outputs zu erzeugen. Zudem wird der Grad an Zufälligkeit erhöht.

Zusammenfassend kann gesagt werden, dass die partielle Füllung der Formen nicht nur die ästhetische Qualität und die Einzigartigkeit des Outputs erhöht, sondern auch den Anwendungsbereich der Methode erweitert. Sie ermöglicht eine adaptivere und vielseitigere Generierung, die für die Zielsetzungen dieser Arbeit besser geeignet ist.

Die ‘chance()’ Funktion ist in allen fertigen Programmen integriert.

In einem Beispiel wurde der Text dieses Abschnitts in Binärcode übersetzt und in die Programme ‘Graustufe’ und ‘RGB’ gegeben.

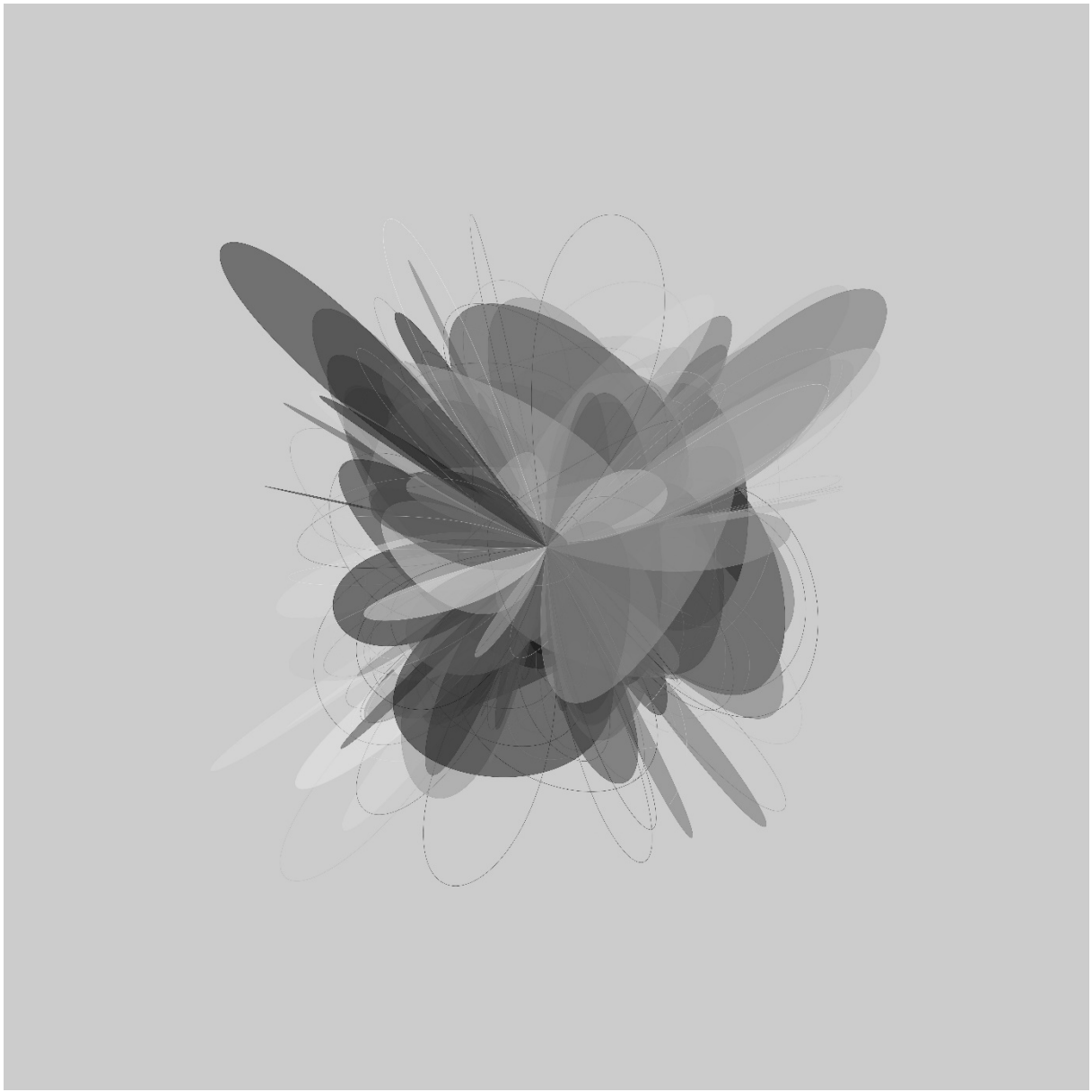


Abbildung 8. Output 3, ‘Graustufe’, ‘if (chance < 0.3) ’

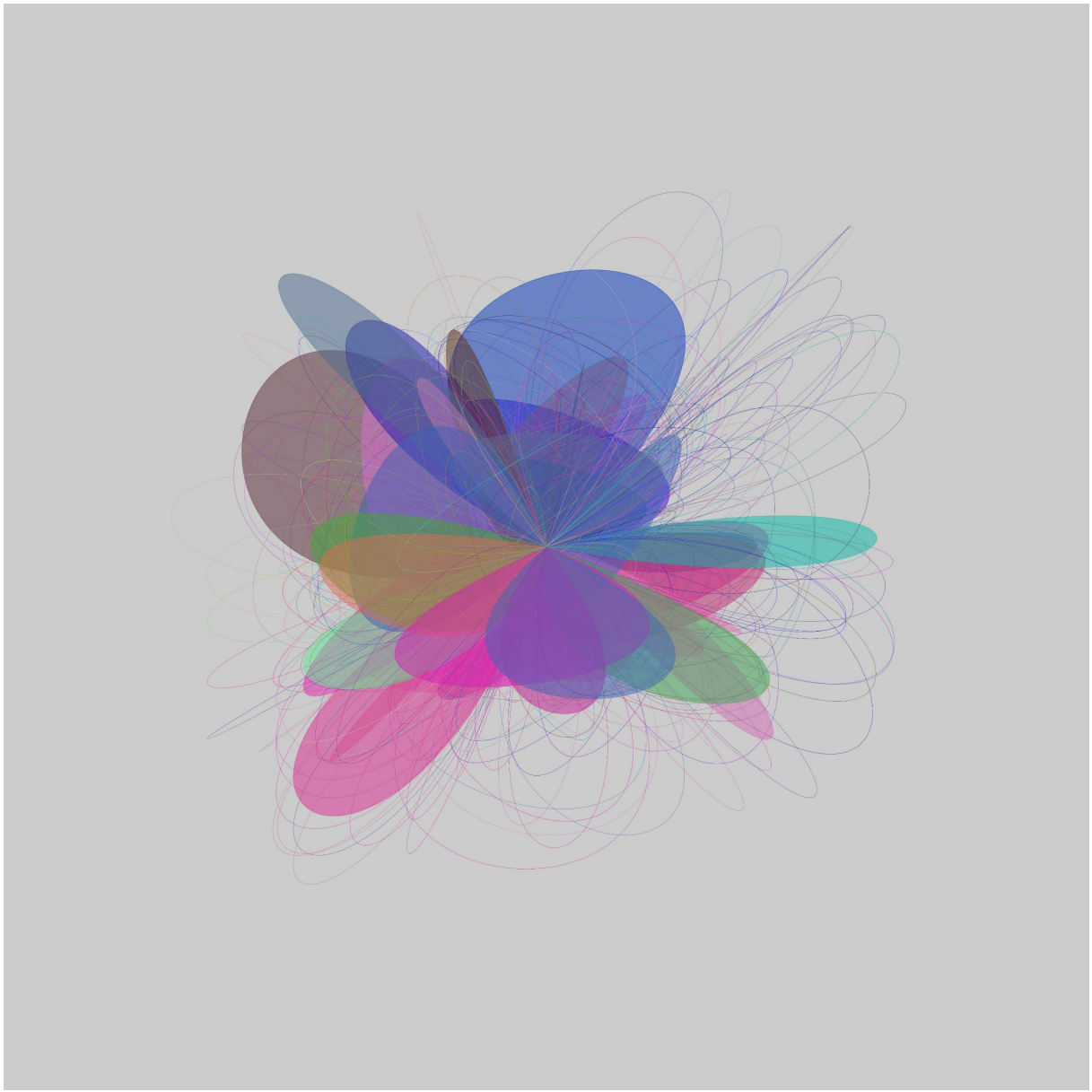


Abbildung 9. Output 4 von 'RGB', 'if (chance < 0.3) '

Verteilt

Bisher: Die Outputs, aus den Programmen ‘Graustufe’ und ‘RGB’ sind mittig ausgelegt und weisen eine Ähnlichkeit auf. In diesen Programmen werden die Start- und Endpunkte der Kurve mittig ausgelegt. cx1, cx2, cy1, cy2 sind verantwortlich für die Krümmung, Start- und Endpunkt sind festgelegt durch Höhe und Breite der Leinwand geteilt durch zwei:

```
bezier(width/2, height/2, x1, y1, x2, y2, width/2, height/2);
```

Neu: Indem man den gemeinsamen Start- und Endpunkt anhand der binären Daten berechnet und diese Position auf der Leinwand wandern lässt , kann die ‘bezier()’ Funktion folgendermassen deklariert werden:

```
bezier(x1, y1, cx1, cy1, cx2, cy2, x1, y1);
```

Die ersten und letzten zwei Stellen entsprechen dem Start- und Endpunkt. So wird nur ein Startpunkt berechnet, basierend auf den binären Daten. Die cx und cy Variablen werden mit dem gleichen Schema berechnet wie in den bisherigen Programmen ‘Graustufe’ und ‘RGB’. Die ‘**draw()**’ Funktion verwendet hier 72- Bit Pakete, weil drei Farbwerte und sechs Punkte berechnet werden, welche je 8-Bit verwenden.

Die Verwendung von Graustufen als weitere Version des Verteilt-Programms mag zwar bestimmte Vorteile haben, wie etwa die Reduzierung der Datenkomplexität, doch im Kontext dieser Arbeit wurden sie als weniger geeignet erachtet. Ein Hauptgrund dafür ist die Einschränkung in der Farbvarianz, die Graustufen mit sich bringen. Durch die Beschränkung auf eine einzige Farbdimension entgehen dem Endprodukt die Möglichkeiten der farblichen Diversifikation und der damit verbundenen Informationstragfähigkeit, die vollfarbige Darstellungen bieten können.

Darüber hinaus sind die Outputs der Graustufe- und RGB-Programme in ihrer Struktur ähnlich, da sie beide die gleichen Parameter für die Steuerung der Bezier-Kurven verwenden. Insbesondere die Verwendung von festen Start- und Endpunkten, die durch die Höhe und Breite der Leinwand geteilt durch zwei festgelegt sind, limitiert die Flexibilität des Generierungsprozesses.

Daher wurde entschieden, die Graustufen-Option für das Verteilt-Programm nicht in das Endprodukt zu integrieren und stattdessen Methoden zu verwenden, die eine grössere Flexibilität in der Farbgebung und Formengenerierung erlauben.

In einem Beispiel wurde der Text dieses Kapitels in Binärcode übersetzt und in das Programm ‘Verteilt’ gegeben.

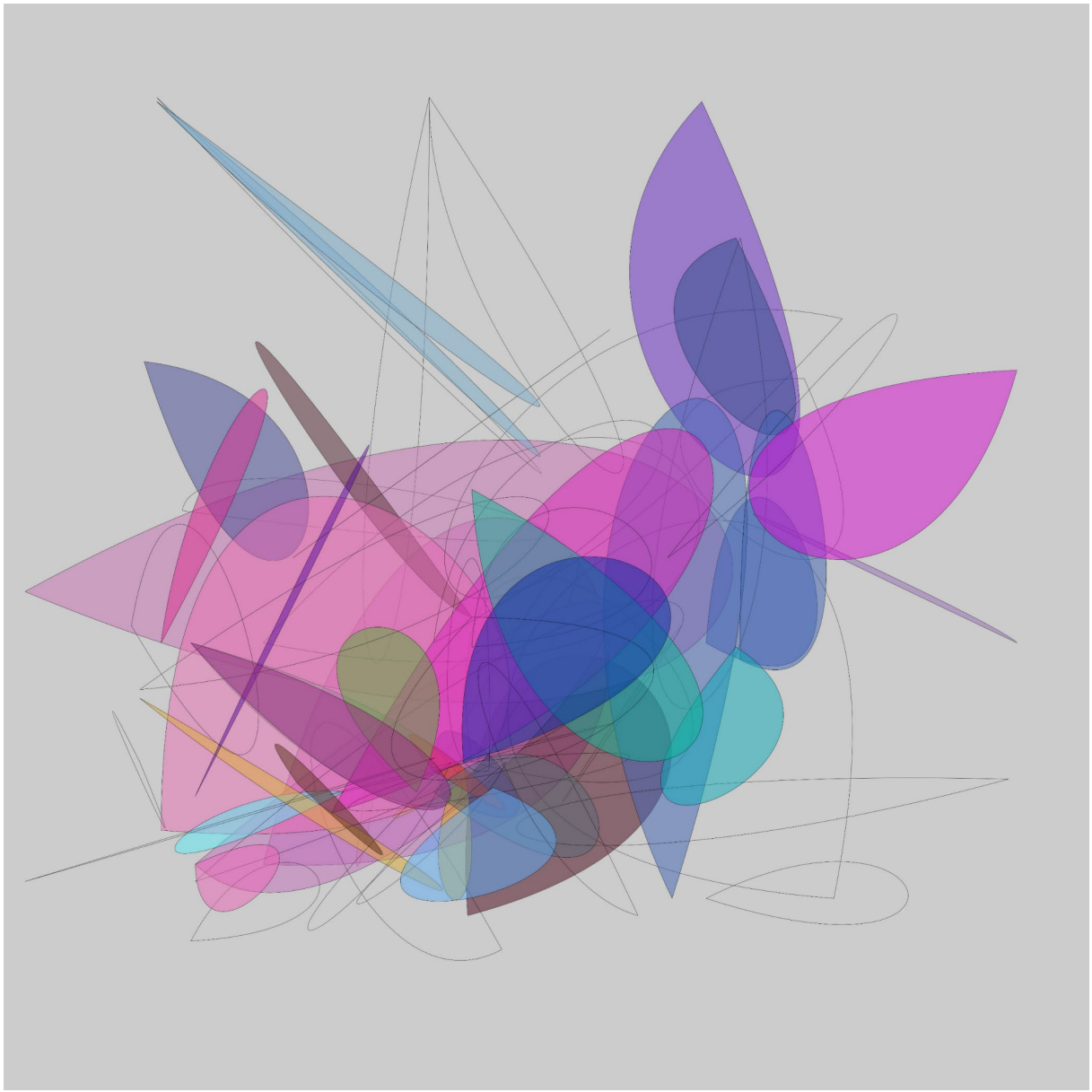


Abbildung 10: Output 5 von 'Verteilt', ‘if (chance < 0.3) ’

Output Analyse

Die Outputs der entwickelten Technologie zeigen eine organische Zusammenstellung von Bézier-Kurven.

Im Graustufen-Programm sind die Kurven in sanften Grautönen gehalten, die sich ineinander verschmelzen und eine beruhigende, fast meditative Atmosphäre erzeugen. Die Kurven erzeugen ein harmonisches, abstraktes Muster, das einen starken Fokus auf die Struktur und die Art und Weise legt, wie die einzelnen Elemente miteinander interagieren.

Das Programm RGB, im Gegensatz dazu, ist lebhaft und farbenfroh. Die Kurven strahlen von einem zentralen Punkt aus und überlappen einander, wodurch vielschichtige Farbverläufe und Mischungen entstehen. Die lebendigen Farben vermitteln ein Gefühl von Freude, Lebendigkeit und Energie. Es wirkt weniger abstrakt als die graue Darstellung und erzeugt ein Gefühl von Leichtigkeit und Bewegung.

Zusammen betrachtet zeigen die beiden Programme, wie Farbgebung und Design die emotionale Reaktion eines Betrachters beeinflussen können. Während beide Bilder dieselbe Grundstruktur nutzen, erzeugen sie durch ihre unterschiedliche Farbpalette und -anwendung völlig verschiedene Emotionen und Eindrücke.

Der Output aus dem Programm ‘Verteilt’ stellt ebenfalls eine Mischung aus Bézier-Kurven dar. Die Struktur ist weniger dicht als in den vorherigen Bildern, wodurch einzelne Elemente und ihre Interaktionen klarer erkennbar sind. Die Farbkombinationen und Überlappungen erzeugen einen dreidimensionalen Eindruck. Das Zusammenspiel von transparenten Flächen und den darunterliegenden Strukturen fügt eine zusätzliche Tiefe hinzu. Die klaren Linien und Kurven, die in das Design integriert sind, zeigen die mathematischen und geometrischen Elemente, die in seine Schaffung eingeflossen sind.

‘Verteilt’ strahlt Ruhe und Ordnung aus, mit einem Hauch von Geheimnis, da der genaue Zweck oder die Bedeutung der dargestellten Struktur nicht ersichtlich ist. Es ist ein ausgezeichnetes Beispiel dafür, wie die Kombination von Kunst und Mathematik zu ansprechenden und nachdenklich machenden Designs führen kann.

Zusammengefasst bieten diese Programme einen faszinierenden Einblick in die vielfältigen Möglichkeiten, die sich durch die Kombination von Mathematik und Zufall ergeben. Jedes Bild besitzt seine eigene Ästhetik und Botschaft, wobei alle durch ihre einzigartige Anwendung von Form, Farbe und Struktur beeindrucken.

Zufallsanalyse

Die Analyse fokussiert sich auf die drei Programme, die binäre Daten verwenden, um Bézier-Kurven zu zeichnen. Alle verwenden einen Seed-Wert für den Zufallszahlengenerator, der aus den binären Daten generiert wird. Ein wesentlicher Aspekt ist die variable Wahrscheinlichkeit (vgl.chance), die in jedem Programm für die Anwendung von Transparenz und Farben eingesetzt wird. Diese Wahrscheinlichkeit kann zwischen 0 und 1 variieren, was einen signifikanten Einfluss auf die Zufälligkeit hat.

Analyse der einzelnen Programme:

Das ‘Graustufe’ Programm setzt 40 Bits pro Kurve ein und beschränkt sich auf Graustufen für die Farbauswahl. Durch die geringere Bit-Anzahl und die Beschränkung auf Graustufen wird die Zufälligkeit im Vergleich zum ‘RGB’ Programm limitiert. Allerdings bietet auch hier die variable Wahrscheinlichkeit für Transparenz und Füllung die Möglichkeit, die Zufälligkeit in bestimmten Aspekten zu erhöhen oder zu verringern.

Im Gegensatz dazu bietet das ‘RGB’ Programm eine breitere Farbpalette. 56 Bits werden pro Kurve verwendet, um die Form und die Farbe der Bezier-Kurven zu steuern. Durch die Verwendung einer höheren Bit-Anzahl wird eine höhere Formvarianz erzielt. Ausserdem wird die Farbauswahl durch die Nutzung von RGB-Werten diversifiziert. Ein wichtiger Aspekt ist die variable Wahrscheinlichkeit für die Anwendung von Transparenz, die es ermöglicht, die Zufälligkeit je nach Bedarf zu steuern.

‘Graustufe’ und ‘RGB’ haben einen zentralen Start- und Endpunkt, was wiederum auf eine geringere Formvarianz zurückgeht. Das ‘Verteilt’ Programm verwendet die höchste Anzahl an Bits pro Kurve, nämlich 72, ausserdem sind Start- und Endpunkt auf der Zeichenfläche verteilt, was zu einer erhöhten Formvarianz führt. Die Farbauswahl ist in diesem Programm durch RGB-Werte bestimmt, was die Zufälligkeit in diesem Aspekt erhöht. Die variablen Wahrscheinlichkeiten für die Anwendung von Farben bieten jedoch Raum für eine Anpassung der Zufälligkeit.

The Road Not Taken

In diesem Kapitel werden die Ansätze dargestellt, welche zwar entwickelt wurden, jedoch nicht zum Endprodukt gehören.

Bilder aus Text dreidimensional zu generieren wäre sehr spannend und böte vermehrte Anwendungsmöglichkeiten. Jedoch stellt sich die Situation gleich wie am Anfang von Manfred Mohrs Karriere: Der Autor besitze kein Gerät, welches solche Medien generieren kann. Ausserdem ist dieser Ansatz in der Umsetzung ungleich komplexer als der gewählte.

In der Entwicklungsphase der Programme, die zum Endprodukt gehören, wurden unzählige Versionen erstellt, welche meistens fehlerhaften oder nicht ansprechenden Output generierten. Im Anhang sind einige dieser Versionen beschrieben, die mit dem Konzept einher gehen: Text in Binär-Code zu übersetzen und diesen reproduzierbar darzustellen mittels Zufallsoperationen.

Block Array

Im Rahmen dieser Arbeit galten ‘Block Array’ Programme als anfänglicher Ansatz. Diese Programme sind erstellte Algorithmen die Ellipsen, Quadrate und Rechtecke mittels des binären Inputs anordnen.

Die Verwendung von 'Block Array' Programmen erwies sich zwar als interessanter Ansatzpunkt, jedoch wiesen sie signifikante Limitationen auf, die ihre Integration in das Endprodukt unpraktikabel machten. Die Variabilität des Outputs dieser Algorithmen ist gering und eignet sich nicht als Methode, die Daten auf einzigartige Weise visuell zu repräsentieren. Sie generieren scheinbar wiederholbare Muster. Zweitens ist die Strukturierung der Formen—Ellipsen, Quadrate und Rechtecke—stark begrenzt und bietet wenig Raum für Diversifikation oder Weiterentwicklung.

Aus diesen Gründen wurde entschieden, 'Block Array' Programme nicht in das Endprodukt zu integrieren, sondern nach alternativen Methoden zu suchen, die eine grössere Flexibilität und Anpassungsfähigkeit bieten. Diese Alternativen sollten auch in der Lage sein, komplexere Outputs zu erzeugen, die für die Zielsetzung dieser Arbeit besser geeignet sind. In diesem Fall betrifft dies den Bézier-Kurven-Ansatz. Der Code zu diesen Programmen findet sich in der GitHub Verlinkung im Anhang.

Auf den folgenden Seiten sind drei Beispiele der oben genannten ‘Block Array’ Programme zu finden, basierend auf dem Text dieses Abschnitts.

Animationen

Ein weiterer Ansatz, welcher zwar entwickelt aber nicht ins Endprodukt eingeflossen ist, sind Animationen. Das erstellte Programm funktioniert auf der Basis von ‘Graustufe’. Jede Kurve wird nacheinander gezeichnet und als Bild zwischengespeichert. Am Schluss werden alle Bilder mit bestimmter Geschwindigkeit abgespielt - ähnlich wie bei einem Stop-Motion Film. Dieses Programm heisst ‘Gifmaker’. Die Animation ist auf der Webseite, auf der Concept Sektion zu finden. Code dazu findet sich in der Verlinkung im Anhang.

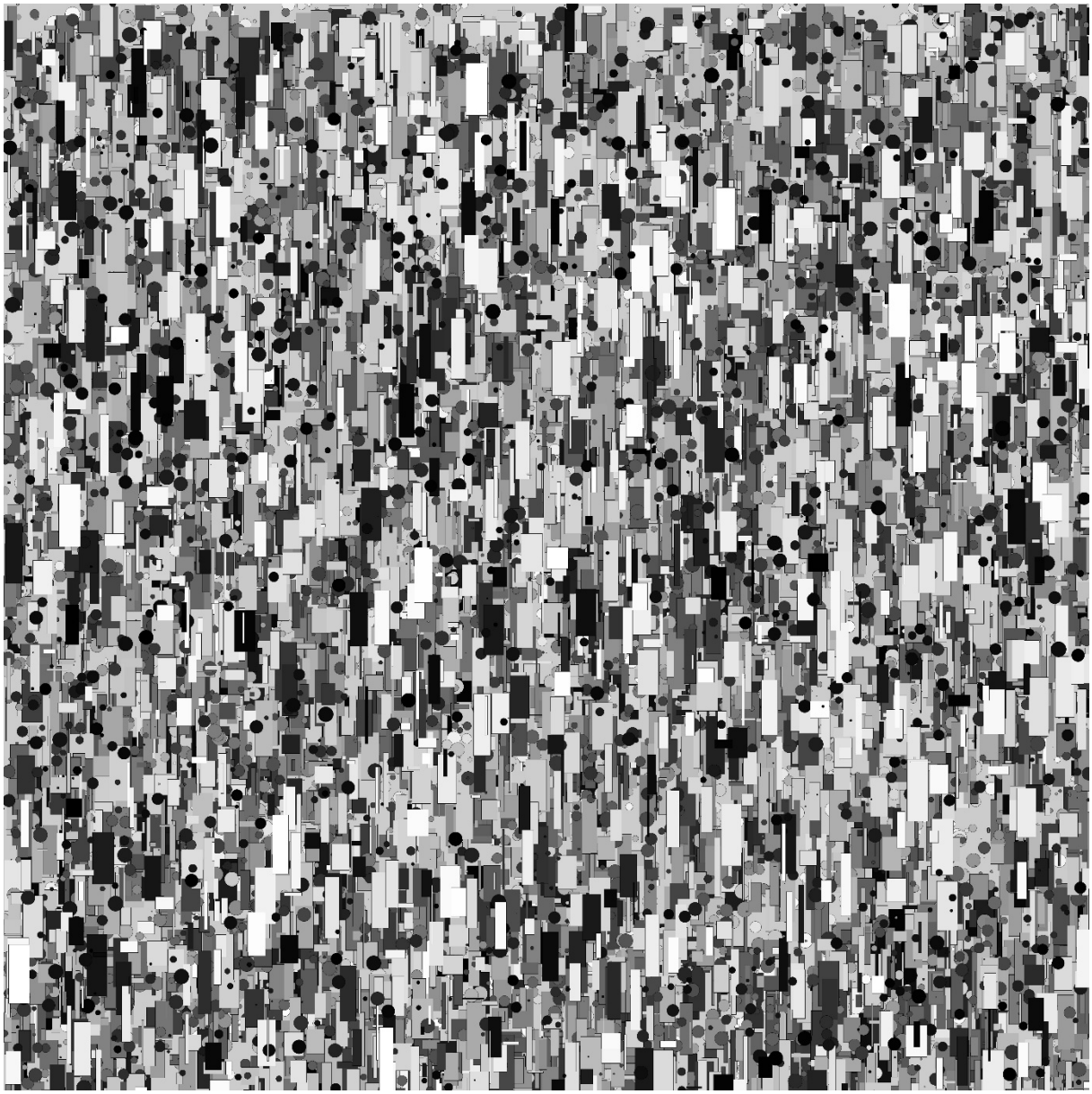


Abbildung 11: Output1 aus ‘Blockarray1’

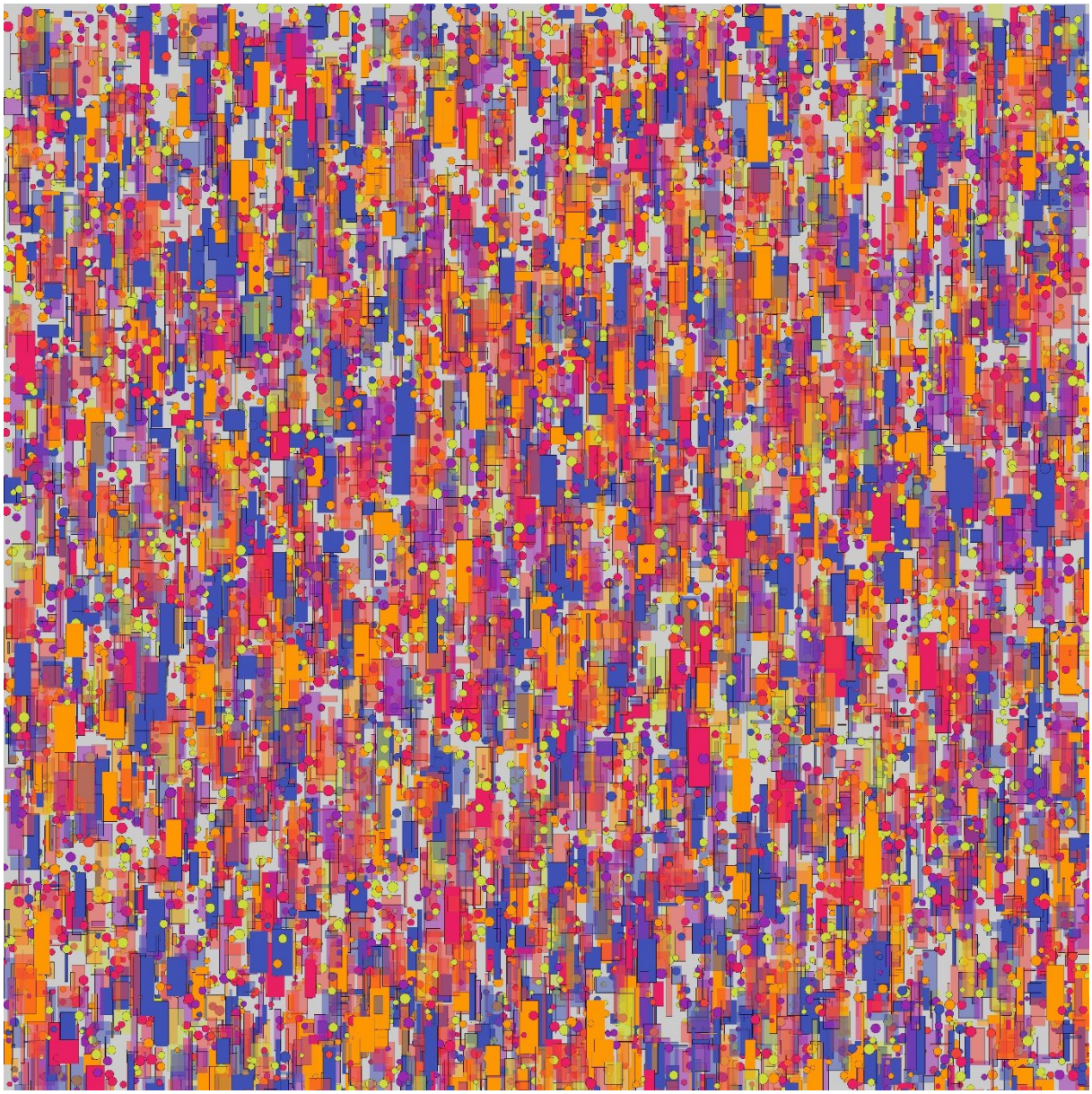


Abbildung 12: Output2 aus ‘Blockarray2’

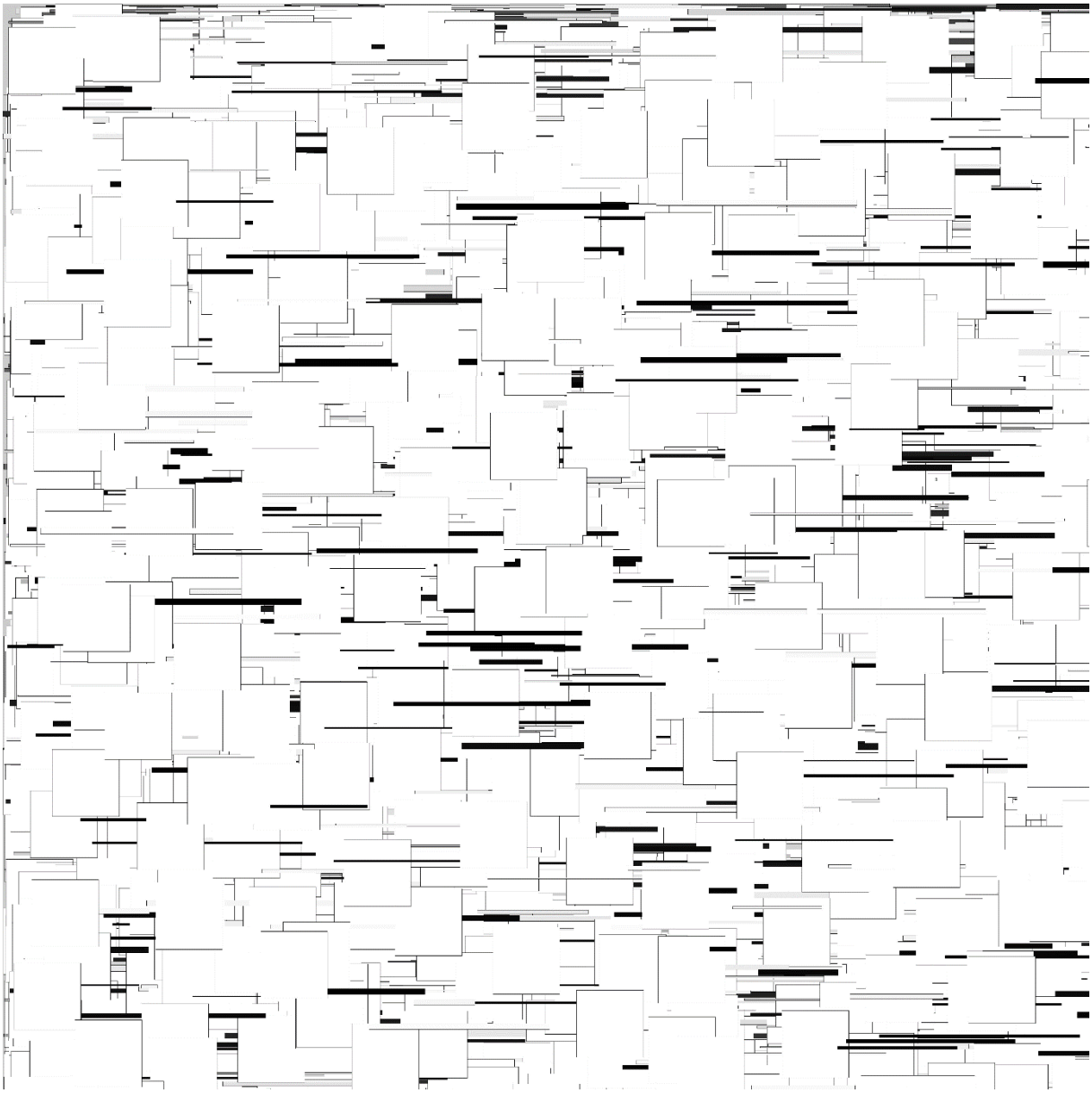


Abbildung 13: Output3 aus ‘Blockarray3’

Endprodukt

Wie in der Einleitung erklärt, stellt das Endprodukt das Ergebnis dar einer tiefgreifenden Auseinandersetzung mit den Möglichkeiten der Digitalisierung in der Kunstszene. Es besteht aus drei miteinander verflochtenen Hauptkomponenten: einer Webseite namens «binarymelodies.com», einer darauf präsentierten NFT-Kollektion und der entwickelten Technologie als Methode. Sie sind so konzipiert, dass sie ein ganzheitliches, digital-vernetztes Erlebnis bieten, das sowohl informativ als auch ästhetisch ansprechend ist. Durch die Integration von Blockchain-Technologie, künstlerischer Kollaboration und Web-Design wird ein Mehrwert geschaffen, der weit über die Summe der einzelnen Teile hinausgeht.

Die eingesetzte Technologie ist in der Lage, Texte in visuelle Darstellungen umzuwandeln. Obwohl die Technik grundsätzlich mit jeglicher Textart arbeiten kann, wurde entschieden, in diesem Projekt Liedertexte von Künstlern aus der Stadt Biel zu verwenden. Dadurch erhält das Projekt einen lokalen Bezug.

binarymelodies.com

Die Webseite «binarymelodies.com» dient als zentrale Schnittstelle und Präsentationsplattform für das gesamte Projekt. Sie fungiert als Informationsportal und schafft einen Raum, in dem die verschiedenen Aspekte des Projekts interagieren können.

Auf der Webseite sind die Outputs der im Projekt eingesetzten Technologien, das zugrunde liegende Konzept sowie Verlinkungen zu externen Plattformen wie *Opensea* und *GitHub* zu finden. Die klare und minimalistische Struktur und Navigation ermöglichen es den Benutzern, sich intuitiv zurechtzufinden. Die Webseite stellt ebenfalls die Künstler Buds und Motis vor, die aus der Stadt Biel stammen. Ihre Werke liefern den Liedtext, der als Input für die verschiedenen Technologiekomponenten des Projekts verwendet wird. Ihre Beteiligung fügt eine zusätzliche Dimension der Authentizität und Lokalität hinzu.

Unten eine Bildschirmaufnahme der Homepage:

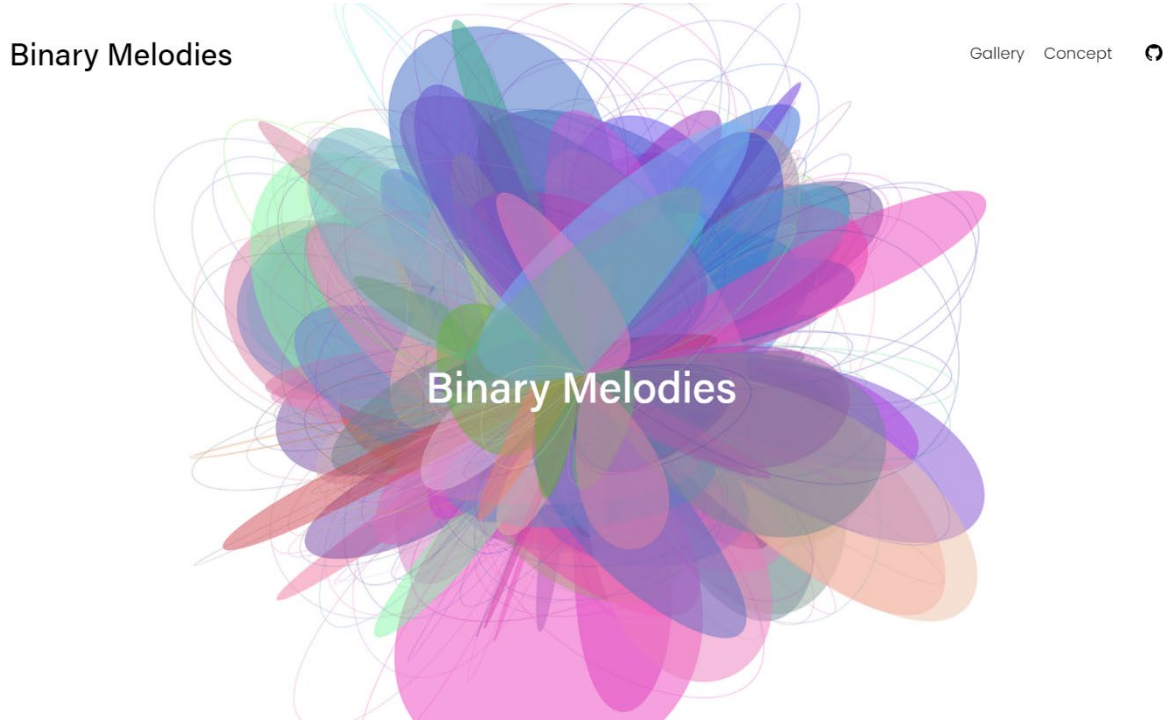


Abbildung 14: Bildschirmaufnahme binarymelodies.com

NFT-Kollektion

Die NFT-Kollektion ist ein wesentlicher Bestandteil des Projekts (vgl. Kapitel Zukunft) und erweitert dessen digitale Präsenz in den Bereich der Blockchain-Technologie. Sie umfasst zehn Lieder, die jeweils in drei verschiedenen Versionen existieren: Graustufe, RGB und verteilt. So entsteht eine Serie von 30 Bildern.

Jedes der zehn Lieder ist in den oben genannten drei Versionen verfügbar, um unterschiedliche ästhetische Erlebnisse zu bieten. Jedes Bild wird im Format 5000 auf 5000 Pixel generiert und hat einen Hintergrund in Graustufe. Alle Bilder werden zudem mit der gleichen Wahrscheinlichkeit für die Füllung berechnet- 30%. Diese Vielfalt ermöglicht es, ein breiteres Publikum anzusprechen und verschiedene Geschmacksrichtungen zu bedienen.

Eine entscheidende Rolle im Verkauf und der Verbreitung der NFT-Kollektion spielen die Marketingaktivitäten, insbesondere auf sozialen Netzwerken. Die beteiligten Künstler 'Buds' und 'Motis' nutzen ihre Online-Plattformen, um auf die Kollektion aufmerksam zu machen. Durch gezielte Posts und das Teilen von Inhalten wird eine grössere Reichweite geschaffen. Dies trägt nicht nur zur Popularität der Kollektion bei, sondern stärkt auch die gesamte digitale Präsenz des Projekts.

Passepartout Biel/ Bienne

Neben den ästhetischen und technologischen Aspekten hat dieses Projekt auch eine soziale Komponente, die durch eine Partnerschaft mit der sozialen Organisation *Passepartout Biel/Bienne* zum Ausdruck kommt. Die Einnahmen aus dem Verkauf der NFT-Kollektion werden an diese Organisation gespendet. Die Spenden unterstützen die sozialen und kulturellen Programme von *Passepartout Biel/Bienne* und tragen zur Förderung eines breiteren gesellschaftlichen Engagements bei. Diese Spendenaktion spiegelt das Bestreben des Projekts wider, einen positiven Einfluss auf die Gemeinschaft auszuüben und darüber hinaus einen Mehrwert zu schaffen. Es erhöht auch die soziale Reichweite und das Bewusstsein für das Projekt, indem es potenzielle Käufer und Interessenten mit einer zusätzlichen Motivation zur Teilnahme animiert. Die NFTs werden auf der Handelsplattform *Opensea* versteigert.

Die Webseite und die NFT-Kollektion werden am 24.10.2023 veröffentlicht.

Die Verlinkung dazu ist im Anhang zu finden.

Schlussfolgerung

Diese Arbeit stellt eine Verschmelzung unterschiedlicher Ansätze in der algorithmischen Kunst dar. Sie kombiniert die systematische Genauigkeit, wie sie in Manfred Mohrs Werk zu sehen ist, mit der dynamischen Vielfalt von Roman Verostkos Kunst. Darüber hinaus baut die Arbeit auf dem modularen Design von Manuel Barbadillo auf. Im Gegensatz zu Barbadillos eher handwerklicher und menschenzentrierter Herangehensweise nutzt diese Arbeit datengesteuerte Methoden.

Mit dieser Arbeit wird nicht nur ein Beitrag zur Weiterentwicklung des Feldes der algorithmischen Kunst geleistet, sondern auch die Forschungsfrage klar und fundiert beantwortet: «Wie lässt sich der Zufallsprozess als Methode zur Transformation eines Textes in ein visuelles Medium effektiv implementieren?» Sie zeigt, dass die algorithmische Kunst eine vielfältige und faszinierende Disziplin ist, die viel Raum für zukünftige Forschungen und Entwicklungen bietet

Roman Verostko Schaffen umfasst Elemente des Zufalls und der Unvorhersehbarkeit. Im Verteilt Programm spiegelt sich die Variabilität der Startpunkte der Bézier-Kurven. Diese Herangehensweise schafft eine künstlerische Dynamik, die jedes Werk einzigartig macht. Im Gegensatz zu Verostko, zu dessen Werk auch textuelle und animatorische Elemente gehören, bleibt der Fokus hier ausschliesslich auf dem statisch visuellen Bereich.

Das zentrale Ergebnis der Arbeit ist die erfolgreiche Anwendung des Zufalls als Methode zur Transformation von Text in visuelle Medien. Hierbei werden Schweizer Musiktexte in eine binäre Sprache übersetzt und anschliessend in visuelle Formen, insbesondere Bézier-Kurven, transformiert. Durch eigens entwickelte Algorithmen und Programme wird eindrücklich demonstriert, dass der Zufall als Schlüsselelement dient für die Transformation von Text zu einzigartigen visuellen Kunstwerken.

In jedem der drei Programme (Graustufe, RGB, Verteilt) ist die Zufälligkeit nicht nur ein technisches, sondern auch ein künstlerisches Merkmal. Durch die Manipulation der Wahrscheinlichkeit ‘chance()’ lässt sich der Grad der Zufälligkeit steuern. Die Technologie bietet hier eine Plattform, auf der die Unvorhersehbarkeit gefeiert und als wichtiger Bestandteil des künstlerischen Ausdrucks anerkannt werden kann. Die Analyse zeigt, dass die technischen Unterschiede zwischen den Programmen nicht nur Auswirkungen auf die Funktionalität, sondern auch auf die künstlerischen Möglichkeiten haben, und unterstreicht die Bedeutung der Zufälligkeit und der variablen Wahrscheinlichkeit als kritische Elemente für die Erkundung von digitaler Kunst.

Diskussion

Wie aus dieser Arbeit hervorgeht, setzt die Generierung solcher Werke Fachwissen voraus. Dieses zu erarbeiten ist aufwendig. Man wird davon ausgehen dürfen, dass Interessentinnen und Interessenten für solche Werke sich in der Regel nicht mit dem notwendigen Programm technischen vorarbeiten befassen wollen. Für diese Kunstinteressenten sollte ein benutzerfreundliches Werkzeug am Markt erhältlich sein.

Dieses kann in Form einer Webseite erfolgen, wo ein entsprechendes Werkzeug aufgerufen werden kann. In diesem können Interessenten ihren Text eingeben und das entsprechende Werk generieren. Die Benutzeroberfläche müsste so gestaltet werden, dass der Benutzer oder die Benutzerin von der Programmierung abgeschirmt wäre. Die Form des Outputs ist vom Benutzerinput vorgegeben , ergänzend können verschiedene Details parametrisiert werden.

Folgende Parameter könnten einstellbar sein:

- Bildgrösse
- Hintergrund
- Farbpalette
- Liniendicke
- Füllungen
- Wahrscheinlichkeit für die Füllungen

Mit solchen Erweiterungen kann zudem ein breiteres Publikum erreicht werden. So können Kunstinteressierte ohne die sonst notwendigen technischen Kenntnisse angesprochen werden: eigentlich alle, die an Form und Farbe interessiert sind.

Redlichkeitserklärung

Ich erkläre hiermit, dass ich die vorliegende Maturaarbeit eigenständig und ohne unerlaubte fremde Hilfe erstellt habe und dass alle Quellen, Hilfsmittel und Internetseiten wahrheitsgetreu verwendet wurden und belegt sind.

Biel, 24.10.2023

Alix Franck

Alix Franck

Literaturverzeichnis

1. 101 Blockchains. (o.J.). Abgerufen von [101blockchains.com](https://101blockchains.com/art-blocks-nft/) (Abgerufen: 19.10.2023).

2. Art Blocks. (o.J.). Abgerufen von [www.artblocks.io](https://www.artblocks.io/info/how-it-works) (Abgerufen: 15.10.2023).

3. AtariArchives.org. (o.J.). Abgerufen von www.atariarchives.org (Abgerufen: 16.10.2023).

4. Cryptopolitan. (2021). Abgerufen von [www.cryptopolitan.com](https://www.cryptopolitan.com/de/ki-und-blockchain-zur-entfesselung-von-web-3-0s/) (Abgerufen: 17.10.2023).

5. DAM. (2021). Abgerufen von [dam.org](https://dam.org/museum/artists/phase_1/) (Abgerufen: 12.10.2023).

6. DappRadar. (o.J.). Abgerufen von [dappradar.com](https://dappradar.com/blog/what-is-the-art-blocks-nft-marketplace) (Abgerufen: 10.10.2023).

7. Database of Digital Art. (o.J.). Abgerufen von dada.compart-bremen.de (Abgerufen: 18.10.2023).

8. Galería Rafa. (o.J.). Abgerufen von www.galeriarafaelortiz.com (Abgerufen: 14.10.2023).

9. GeeksforGeeks. (o.j.). Abgerufen von [GeeksforGeeks](https://www.geeksforgeeks.org/program-binary-decimal-conversion/) (Abgerufen: 25.10.2023).

10. Greenberg, I. (2007). Abgerufen von [Processing: Creative Coding and Computational Art](https://www.friendsofed.com) (Abgerufen: 4.10.2023).

11. Knuth, D. E. (1997). The Art of Computer Programming (vol. 2: Seminumerical Algorithms) (3rd ed.).

12. Mohr, M. (2022). Abgerufen von [emohr.com](http://www.emohr.com/ww1_out.html) (Abgerufen: 20.10.2023).

13. Processing 3.0 Documentation. (o.J.). Abgerufen von [Processing 3.0 Documentation](https://processing.org/reference/randomSeed_.html) (Abgerufen: 8.10.2023).

14. Processing Foundation. (2021). Abgerufen von [Processing Official Website] (https://processing.org/reference/random_.html) (Abgerufen: 6.10.2023).

15. Reas, C. & Fry, B. (2014). Abgerufen von [Processing: A Programming Handbook for Visual Designers and Artists] (https://mitpress.mit.edu/books/processing) (Abgerufen: 24.10.2023).

16. Spalter Digital Art on Manfred Mohr. (o.J.). Abgerufen von [spalterdigital.com](https://spalterdigital.com/artists/manfred-mohr/) (Abgerufen: 24.10.2023).

17. Victoria and Albert Museum Collection on Manuel Barbadillo. (o.J.). Abgerufen von [collections.vam.ac.uk](https://collections.vam.ac.uk/item/O152188/adfera-print-barbadillo-manuel/) (Abgerufen: 16.10.2023).

18. Verostko, R. (o.J.). Abgerufen von [verostko.com](http://www.verostko.com/resume-all.html) (Abgerufen: 2.10.2023).

Abbildungsverzeichnis

Anhang

Im Anhang werden Inhalte aufgelistet, die nicht direkt zur Arbeit gehören.

Verlinkung zu allen Inhalten des Endprodukts:

Webseite

https://binarymelodies.com/

NFT-Kollektion

https://opensea.io/collection/binarymelodies

Code zu den Programmen

https://github.com/Alyxfranck/Binarymelodies

Verlinkung zu den Programmen aus The Road Not Taken:

https://github.com/Alyxfranck/Binarymelodies/tree/The-Road-Not-Taken

Verständnis Verweise

Generative Kunst: https://de.wikipedia.org/wiki/Generative_Kunst

NFT: https://de.wikipedia.org/wiki/Non-Fungible_Token

Blockchain: https://de.wikipedia.org/wiki/Blockchain

Künstliche Intelligenz: https://de.wikipedia.org/wiki/KI-Kunst

Hashfunktionen: https://de.wikipedia.org/wiki/Hashfunktion

Processing Dokumentation: https://processing.org/reference

Frühere ‘Verteilt’ Version: Verteilt Beta

Diese Version berechnet die Krümmung der Kurven anhand der binär Daten, Start- und End Punkt wird durch die ‘random()’ Funktion bestimmt.

Anfangsversion des ‘Graustufen’ Programms: Graustufe Beta

Der Startpunkt wurde hier mittig festgelegt, der Rest der Kurve wurde mit einer ‘random()’ Funktion berechnet.