Diplomarbeit

Konzeption und prototypische Realisierung einer Software für die Informationsvisualisierung

Benjamin Knofe

13.10.2011

Betreuer:
Professor Dr.-Ing. Robert Müller

Hochschule für Technik, Wirtschaft und Kultur Leipzig

Danke
für Kritik, Kommentare, Diskussionen,
Korrekturen, Code und Hilfe
(in alphabetischer Reihenfolge)

Theresia Becher
Pyry Jahkola
Stephan Keller
Johannes Knofe
Moritz Kreutzer
Ted Mosby
Professor Dr.-Ing. Robert Müller
Christina Sanko
Sarah Walter
Philip Whitfield
René Zschoch

Die Diplomarbeit

„Konzeption und prototypische Realisierung
einer Software für die Informationsvisualisierung“

von Benjamin Knofe (http://www.videosynthesis.net)

Inhaltsverzeichnis

1 Einleitung

Die Menge an digital gespeicherten Daten nimmt stetig zu. Im Jahr 2011 werden laut einer neuen Studie[1] voraussichtlich 1,8 Zettabyte[2] an Daten erzeugt und kopiert. In nahezu allen Bereichen unseres täglichen Lebens werden Daten erhoben, gemessen und gespeichert. Neben der Wissenschaft, Wirtschaft und Kultur verfolgen öffentliche Bereiche der Politik und Verwaltung eine neue Strategie im Umgang mit Daten. So haben Städte, wie beispielsweise Leipzig, eine öffentliche Datenschnittstelle. Staaten, wie Großbritannien, verfolgen ein ganzheitliches Konzept des e-Government. Aufgrund dieser aktuellen Entwicklungen ist es notwendig, neue Strategien zu entwickeln, die diese Daten für die jeweiligen Zielgruppen sinnvoll extrahieren, aufarbeiten und darstellen. Aus diesem Grund wurden in verschiedenen Forschungsfeldern, wie beispielsweise Statistik, Data-Mining, Computergrafik und Informatik spezielle Lösungsansätze entwickelt. Die Informationsvisualisierung verbindet Bereiche dieser Disziplinen in einem eigenständigen Forschungsgebiet. In diesem wird die Frage geklärt, wie immer abstrakter und komplexer werdende Daten in einer für den Menschen adäquaten Weise aufbereitet und dargestellt werden können. Informationen in Form von Mustern, Wiederholungen und Anomalien sind

[1] Vgl. (EMC, 2011)

[2] 1 Zettabyte = $1 \cdot 10^{21}$ Byte = 1.000.000.000.000 Gigabyte

dadurch leichter auffindbar und erlauben Rückschlüsse auf den eigentlichen Forschungsgegenstand. Ursprünglich wurden Visualisierungstechniken nur im naturwissenschaftlichen Kontext angewendet. Im Gegensatz dazu hat sich die Informationsvisualisierung zu einem populären Arbeitsfeld entwickelt. Sie versucht ein breites Spektrum an Problemen zu lösen und ist nicht mehr nur ein Werkzeug von Experten und Wissenschaftlern. Alltägliche Anwendungen im Bereich der sozialen Netzwerke, beim Onlineshopping und bei der Suche nach Informationen im Internet können mit der Informationsvisualisierung verbessert werden. Sie unterstützt die Gestaltung von Benutzerschnittstellen, um diese verständlicher und intuitiv einsetzbar zu machen. Damit wird es vor allem Gelegenheitsanwendern leicht gemacht, Informationen und Wissen noch besser mit Anderen zu kommunizieren und zu teilen. Das erfordert (digitale) Werkzeuge, die den Benutzer unterstützen und ihm die Möglichkeit geben, Daten für Problemlösungen nutzen zu können. Um die Informationsvisualisierung erfolgreich einzusetzen, muss eine Vielzahl von Fragestellungen beantwortet und gelöst werden.

1.1 Problemstellung

Abbildung 1.1 zeigt beispielhaft eine Informationsvisualisierung, in der die unterschiedliche Bedeutung von Farben in verschiedenen Kulturkreisen visualisiert wird. Die einzelnen Ringe der kreisförmigen Darstellung bilden zehn unterschiedliche Kulturkreise ab. Der Kreis ist unterteilt in 84 Begriffe. Der Farbton der Rechtecke stellt dar, welche Farbe einem Begriff von Menschen einer bestimmten Kultur zugeordnet wird. Sie wurde von David McCandless

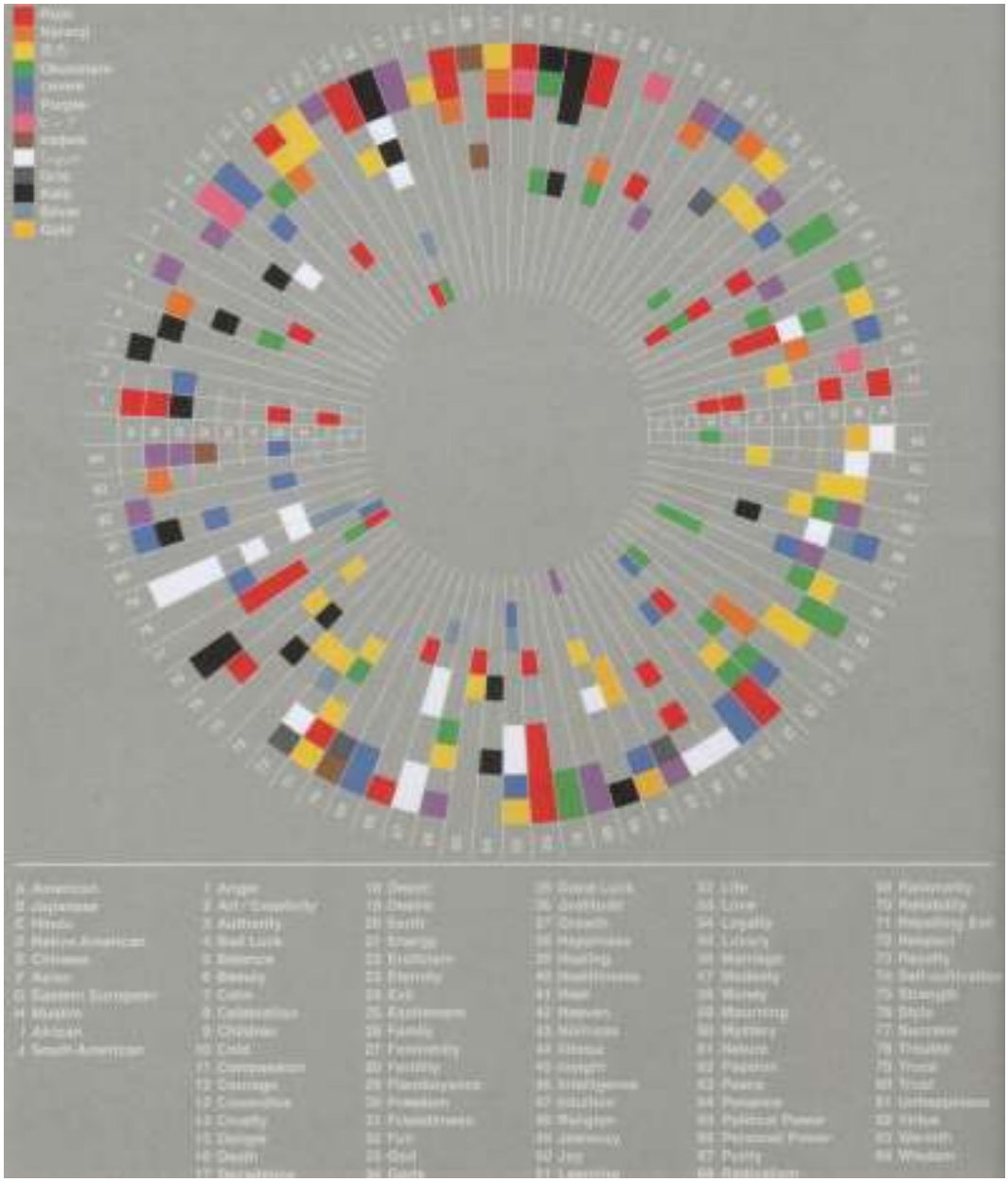


Abbildung 1.1: Beispiel einer Informationsvisualisierung aus (McCandless, 2009, S. 76)

mit einem Vektorgrafikprogramm erstellt, um eine tiefgreifende Auseinandersetzung zwischen ihm (als menschlichen Betrachter) und den Informationen zu ermöglichen und Entscheidungen über das Design direkt treffen zu können.[3] Der Nachteil dieser Arbeitsweise ist eine zeitaufwändige manuelle Transkription aller Datenwerte in eine grafische Darstellung. Bei der Visualisierung von komplexeren Zusammenhängen und größeren Datenmengen würde dieser Nachteil noch verstärkt werden und es teilweise unmöglich machen, eine Visualisierung durchzuführen.

In dieser Arbeit wird untersucht, ob es möglich und sinnvoll ist, diesen Prozess zu automatisieren, ohne Einschränkungen für den Anwender festlegen zu müssen. Eine Software muss dafür einen universellen Ansatz verfolgen, um ein breites Spektrum an unterschiedlichsten Bearbeitungsfällen abdecken zu können. Aktuelle Standardsoftware, wie beispielsweise Tabellenkalkulationsprogramme oder Programmierbibliotheken, können das nur teilweise leisten oder beschränken sich auf spezielle Anwendungsgebiete. Meist kann dabei, für die Transkription von Daten zu einer grafischen Darstellung nur aus einer begrenzten Auswahl an Voreinstellungen gewählt werden.

Um eine universelle Software zu konzipieren, die dem Benutzer eine größtmögliche Gestaltungsfreiheit einräumt, sind folgende Vorüberlegungen notwendig:

- Welche Art von Daten gibt es und wie können diese klassifiziert werden?
- Welche Möglichkeiten der grafischen Darstellung gibt es und wie können diese eingesetzt werden?

[3] Vgl. (McCandless, 2011)

- Wie können Daten sinnvoll in eine grafische Darstellung transkribiert werden?

Diese Vorüberlegungen machen ein Softwarekonzept nötig, das den Anwender besonders bei der arbeitsintensiven Transkription von Daten in eine grafische Darstellung unterstützt. Um eine iterative und kreative Benutzung zu ermöglichen, muss dieser Vorgang automatisch durchgeführt werden und unmittelbar eine Ausgabe erzeugen.

Die Konzeption einer auf diese Aspekte spezialisierte Software ist Gegenstand dieser Arbeit.

1.2 Aufbau der Arbeit

Um eine Lösung für das Problem eines universellen Informationsvisualisierungsansatzes zu finden, werden in dieser Arbeit Schritte durchgeführt, welche die Konzeption einer Software zum Ziel haben. Diese Software wird prototypisch implementiert.

In Kapitel 2 werden die grundlegenden Begriffe beschrieben, die für das Arbeitsfeld der Informationsvisualisierung wichtig sind. Es werden die Begriffe *Daten* und *Informationen* definiert und eine Klassifikation von Daten vorgenommen. Nachfolgend wird ein System zur Beschreibung der Grundelemente einer grafischen Darstellung eingeführt.

In Kapitel 3 wird der Begriff der Informationsvisualisierung definiert und von verwandten Arbeitsfeldern abgegrenzt. Danach werden die Bearbeitungsziele der Informationsvisualisierung und die Einteilung des Arbeitsprozesses

beschrieben. Im letzten Abschnitt dieses Kapitels wird die grafische Darstellung als Produkt der Informationsvisualisierung charakterisiert.

In Kapitel 4 wird eine Software konzipiert. Der Aufgabenbereich der Software wird durch Anforderungen definiert. Der nächste Abschnitt beschreibt die visuelle Programmierung als Bedienkonzept. Die Benutzung der Software wird anhand einer Prozesskette dargestellt. Im letzten Abschnitt dieses Kapitels wird auf bestehende Softwarelösungen eingegangen.

Kapitel 5 dokumentiert die Umsetzung des Prototyps. Neben den verwendeten Technologien werden Teile der Software beschrieben und erläutert, welche Funktionen diese haben. Dann wird die Benutzung der Software anhand eines Beispiels gezeigt. Am Ende des Kapitels wird auf Probleme bei der Entwicklung eingegangen.

In Kapitel 6 erfolgt die Auswertung des Prototyps. Dieser wird anhand der zuvor aufgestellten Anforderungen evaluiert.

In Kapitel 7 werden die Ergebnisse dieser Arbeit aufgezeigt und ein Ausblick auf mögliche weiterführende Forschungsarbeit gegeben.

2 Grundlagen

Dieser Teil der Arbeit beschreibt die grundlegenden Begriffe, die einen Zugang zum Arbeitsfeld der Informationsvisualisierung ermöglichen.

2.1 Daten und Information

Daten werden als „zum Zweck der Verarbeitung zusammengefasste Zeichen, die aufgrund bekannter oder unterstellter Abmachungen Informationen (d.h. Angaben über Sachverhalte und Vorgänge) darstellen“[1], beschrieben. Als Gegenstand dieser Arbeit werden digital gespeicherte Daten betrachtet. Diese liegen in einer standardisierten Form vor und können für die elektronische Weiterverarbeitung und Kommunikation genutzt werden. Diese Daten sind dabei formalisierte Repräsentationen einer oder mehrerer Informationen, die übermittelt werden sollen. Daten können dabei aus der realen Welt stammen, beispielsweise erhoben durch die Messung einer physikalischen Größe, oder sie wurden digital erzeugt, beispielsweise durch computergestützte Simulationen oder durch die Nutzung von Telekommunikationstechnik. Der Anwender ist an

[1] (Gabler, 2011)

den Informationen und dem aus ihnen resultierenden Erkenntnisgewinn interessiert, die er mit verschiedenen Techniken aus diesen Daten gewinnen kann.

Der Begriff der Information ist schwer einzugrenzen und wird in unterschiedlichen Wissenschaften in verschiedenen Formen ausgelegt. Daher kann noch nicht von einer einheitlichen Theorie der Information gesprochen werden. Es beschäftigen sich viele verschiedene Gebiete wie die Informatik, Informationstheorie, Nachrichtentechnik und andere mit diesem Begriff.[2] Höher (2011) beschreibt die Information wie folgt:

> „Die Informationstheorie definiert Information als eine quantitativ bestimmbare Wissenzunahme durch die Übermittlung von Zeichen in einem Kommunikationssystem.“[3]

Information ist das Ergebnis der Kommunikation, das beim Empfänger einen Erkenntnisgewinn auslöst, sofern dieser den Informationsinhalt entschlüsseln kann. Dafür müssen sich Sender und Empfänger auf ein Zeichensystem einigen. In Bezug auf grafische Darstellungen definiert Bertin (1974) die Information als der transkribierbare Inhalt eines Gedankens.[4] Das heißt, die Information ist der Teil der Daten, die in einer Informationsvisualisierung dargestellt werden kann. Die Informationsvisualisierung hat die Aufgabe, diese Information für den Menschen aufzudecken und in verständlicher Form aufzubereiten. Die reinen Daten, die zwar diese Informationen enthalten, aber für den menschlichen Betrachter nicht erfassbar sind, werden in ein grafisches Zeichensystem transformiert. Dieses grafische System erlaubt dem Menschen eine natürliche, visuelle und damit effektivere Analyse.

[2] Vgl. (Informationsbegriff, 2011)
[3] (Höher, 2011, S. 3)
[4] Vgl. (Bertin, 1974, S. 13)

2.2 Eigenschaften von Daten

In diesem Abschnitt werden die Eigenschaften von Daten beschrieben. Sie bestimmen, wie die eigentlichen Informationen in den Daten codiert sind. Digitale Daten bilden den Ausgangspunkt für die Konzeption einer Informationsvisualisierung. Die speziellen Eigenschaften von Daten sind maßgeblich für weitere Schritte der Informationsvisualisierung.[5] Schumann und Müller (2000) haben ein Modell zur Beschreibung der Eigenschaften von Daten entwickelt. Jeder Datensatz[6] hat folgende Merkmale, nach denen er klassifiziert werden kann:

Beobachtungsraum Der Beobachtungsraum beschreibt den Raum, in dem Daten erhoben werden. Dieser Raum kann ein dreidimensionaler Raum sein, kann aber auch abstrakt beschrieben werden und nur eine einzige oder mehr als drei Dimensionen besitzen. Alle Dimensionen des Beobachtungsraumes werden als *unabhängige Variablen* bezeichnet, da sie vor der Datenerhebung festgelegt werden und als beschreibende Begriffe für den zu visualisierenden Sachverhalt gelten.[7] Im Beispiel aus Abbildung 1.1 können die gewählten Kulturkreise und die verschiedenen Begriffe als unabhängigen Variablen bezeichnet werden.

Beobachtungspunkt Ein Beobachtungspunkt beschreibt eine Position im Beobachtungsraum an dem Daten vorhanden sind. Die Anzahl und die Verteilung der Beobachtungspunkte sind unabhängig von der Beschaffenheit

[5] Vgl. Abschnitt 3.4

[6] Ein Datensatz ist eine Menge von Daten.

[7] (Bertin, 1974, S. 24) bezeichnet die unabhängigen Variablen als Invariante und definiert: „Die Invariante ist der vollständige und in bezug [sic] auf alle vorgegebenen Begriffe invariable Kennzeichnungsbegriff."

des Beobachtungsraumes. Beobachtungspunkte haben einen Wirkungskreis, der die „räumliche“ Gültigkeit der Werte beschreibt. Dieser kann punktuell, lokal oder global sein. Im Beispiel aus Abbildung 1.1 sind alle farbigen Rechtecke im Kreis Beobachtungspunkte, denn an diesen konnte ein zugehöriger Farbwert recherchiert werden.

Merkmal Das Merkmal ist eine Größe, die an einem Beobachtungspunkt gemessen oder berechnet wird. Dabei kann ein Beobachtungspunkt mehrere Merkmale besitzen. Merkmale werden als *abhängige Variablen* bezeichnet, da sie von der Position des Beobachtungspunktes im Beobachtungsraum abhängig sind. Im Beispiel aus Abbildung 1.1 ist der Farbton eines Rechtecks im Kreis ein Merkmal.

Ausprägung Die Ausprägung beschreibt den eigentlichen Wert, den ein Merkmal annehmen kann. Dieser Wert hat einen Wertebereich und kann ein Skalar oder Vektor sein.[8] In Abbildung 1.1 ist beispielsweise im amerikanischen Kulturkreis (A) für Liebe (53) die Farbe Rot ablesbar. Der dazugehörige Wertebereich der Farben wird im oberen linken Bereich der Abbildung dargestellt.

Die Begriffe Beobachtungsraum, Beobachtungspunkt, Merkmal und Ausprägung dienen zur Klassifikation von Daten und werden als Metadaten bezeichnet.

[8] In einem dreidimensionalen Beobachtungsraum kann einem Beobachtungspunkt als Vektor angesehen werden. Im Falle einer Ausprägung als Vektor, liegt damit an der Position des Beobachtungspunktes im Beobachtungsraum ein weiterer Vektor vor.

2.3 Klassifikation von Daten

Die Klassifikation von Daten wird sehr unterschiedlich durchgeführt. Schumann und Müller (2000) weisen auf eine uneinheitliche Begriffswahl in unterschiedlichen Publikationen hin.[9] Hier wird eine vereinfachte Einteilung dargestellt. Diese wird nach der Anzahl der abhängigen und unabhängigen Variablen und nach dem Ursprung der Daten vorgenommen.

2.3.1 Mehrdimensionale Daten

Ein Beobachtungsraum hat eine endliche Anzahl an Dimensionen. Ist die Anzahl dieser *unabhängigen* Variablen größer Eins, wird von *mehrdimensionalen Daten* gesprochen. Das Beispiel in Abbildung 1.1 hat zwei unabhängige Variablen (Kulturkreis und Begriff) und damit zwei Dimensionen.

Eine Sonderform der mehrdimensionalen Daten sind die raumbezogenen Daten. „Raumbezogene Daten liegen dann vor, wenn der Beobachtungsraum Ortskoordinaten enthält, die ein 2- oder 3-dimensionales räumliches Bezugssystem definieren.“[10] Beispielsweise bedeutet das, dass mindestens zwei und maximal drei unabhängige Variablen des Datensatzes ein Koordinatensystem beschreiben können. Diese Form der Informationsvisualisierung findet häufig in Geoinformationssystemen Anwendung.

In Bezug auf den Beobachtungsraum sind die zeitbezogenen Daten ein weiterer Spezialfall. Dabei ist eine unabhängige Variable mit der physikalischen Größe Zeit assoziiert. Dieser Bezug entspricht nicht immer den tatsächlichen

[9] Vgl. (Schumann und Müller, 2000, S. 171)
[10] (Schumann und Müller, 2000, S. 220)

zeitlichen Änderungen und macht eine adäquate Skalierung der zeitbezogenen Dimension notwendig. Beispielsweise besteht die Möglichkeit, bei einer Wetterstation die kontinuierliche Veränderung der Außentemperatur nur einmal pro Stunde zu messen. Dadurch liegen die Messwerte in diskreten Intervallen vor. Diese Problematik muss bei der Erzeugung der Darstellung beachtet werden.

2.3.2 Multiparameterdaten

Daten werden als *Multiparameterdaten* bezeichnet, wenn die Anzahl der *abhängigen* Variablen größer Eins ist und der Beobachtungsraum nicht vernachlässigt werden kann. Das setzt komplexere Informationsvisualisierungen voraus, da unabhängige und abhängige Variablen gemeinsam dargestellt werden müssen. Die Unterscheidbarkeit zwischen diesen Klassen muss dabei erhalten bleiben. Ein Beispiel für einen Multiparameterdatensatz ist eine Liste mit Wetterstationen, die durch ihre Koordinaten bestimmt sind. Zusätzlich dazu liegen für jede Wetterstation unterschiedliche Daten für Temperatur, Luftdruck und Niederschlag vor.[11]

Eine Sonderform der Multiparameterdaten sind die multivariaten Daten. Dabei kann der Beobachtungsraum vernachlässigt werden, da er keine Rolle bei der Weiterverarbeitung der Daten spielt. Multivariate Daten können in mehrdimensionale Daten überführt werden, wenn für jedes vorhandene Merkmal

[11] Das Beispiel aus Abbildung 1.1 ist kein Multiparameterdatensatz, da nur der Farbton der Rechtecke als abhängige Variable bezeichnet werden kann. Durch zusätzliche Informationen könnten die zugrundeliegenden Daten aber zu einem Multiparameterdatensatz erweitert werden. Beispielsweise könnte die Wichtigkeit der einzelnen Begriffe in einem Kulturkreis in der Größe der Farbfläche dargestellt werden. So würde zum Beispiel der Begriff „Religion“ (48) für bestimmte Kulturkreise größer dargestellt werden. Andere Farbflächen erschienen dagegen kleiner.

eine Dimension in einem neuen Beobachtungsraum aufgespannt wird.[12] Ein Datensatz, der alle Antworten von befragten Personen einer anonym geführten Umfrage enthält, ist ein Beispiel für Multiparameterdaten. Die einzelnen Fragen sind die abhängigen Variablen und die Antworten der Befragten sind deren Ausprägungen. Unabhängige Variablen wie beispielsweise Alter, Wohnort und Geschlecht der Person fehlen dagegen.

2.3.3 Abstrakte Daten

Die wissenschaftlich-technische Datenvisualisierung ist auf naturwissenschaftliche Daten fokussiert. Das können Daten über die menschliche Anatomie, das Universum oder über Moleküle und Atome sein. Diese Daten haben einen räumlichen Bezug. Visualisierungen von Abstraktionen des physikalischen Raumes sind möglich. Trotzdem hat die zugrunde liegende Information eine geometrische Natur.[13] Fehlt ein naturwissenschaftlicher Raumbezug oder ist er nicht sinnvoll, werden Daten als *abstrakte Daten* bezeichnet. Zwischen dem Beobachtungsraum dieser Art von Daten und unserer realen dreidimensionalen Umgebung und unserem zeitlichen Empfinden besteht kein oder nur ein sehr schwacher Zusammenhang. Abstrakte Daten werden meist in rein digitalen und virtuellen Prozessen erzeugt. Als Beispiel können die Datei- und Ordnerstrukturen eines Betriebssystems oder Texte und Quellcode von Programmen genannt werden.

Eine spezielle Form der abstrakten Daten sind strukturelle Beziehungen zwischen Datenobjekten. Diese Daten beschreiben nicht Beobachtungspunkte und

[12] Vgl. (Schumann und Müller, 2000, S. 172)
[13] Vgl. (Card u. a., 1999, S. 6)

deren Merkmale, sondern die Beziehungen zwischen diesen Punkten und ihren abhängigen und unabhängigen Variablen. Unterschieden wird dabei zwischen hierarchischen Formen und Netzwerken. Beispiele sind die Relationen zwischen Dokumenten, Medien und Personen in einem Netzwerk.[14]

Rein zeitbezogene Daten werden in dieser Arbeit den abstrakten Daten zugeordnet. Ein Börsenkurs ist bei näherer Betrachtung abstrakt, da kein räumlicher Bezug stattfindet. Die einfache Abbildung in einem Diagramm, bei dem die Zeit auf der Abszissenachse und der Wert einer Aktie auf der Ordinatenachse abgebildet sind, sind nur erlernte Metaphern. Die „Höhe“ des Aktienwertes stellt dabei keinesfalls eine räumliche Höhe dar, sondern eine abstrakte Wertebeschreibung.

Die Informationsvisualisierung, wie sie in Kapitel 3 beschrieben wird, benutzt vorrangig *abstrakte Multiparameterdaten* als Grundlage einer grafischen Darstellung.

2.4 Datentypen

Datentypen beschreiben die eigentlichen Ausprägungen eines Merkmals und können zu einer zusätzlichen Klassifikation beitragen. Das ist wichtig, um erkennen zu können, welche Verarbeitungschritte mit den Daten möglich sind. Meist erfolgt eine Einteilung in drei Klassen. Während Bertin (1974) von qualitativen, geordneten und quantitativen Komponenten spricht, unterscheiden Preim und Dachselt (2010) nominale, ordinale und quantitative Datentypen. Hier wird die neuere Einteilung von Preim und Dachselt (2010) erläutert.[15]

[14] Vgl. (Preim und Dachselt, 2010, S. 448ff)
[15] (Preim und Dachselt, 2010, S. 449)

Nominale Datentypen Meist werden Namen und Bezeichnungen von Datenobjekten als nominale Datentypen klassifiziert. Ihr Abstand ist äquidistant, da sich vor einer eingehenden Analyse keine Ausprägungen besonders nah oder entfernt voneinander zeigen. Die einzige mögliche Operation ist der Test auf Gleichheit beziehungsweise Ungleichheit. Es gibt keine allgemeingültige, eindeutige Reihenfolge nach der die Ausprägungen sortiert werden können. Diese kann nur nach bestimmten Gesichtpunkten erfolgen. Das heißt, eine künstlich erzeugte Ordnung wird eingebracht, zum Beispiel durch alphabetische Sortierung. In Abbildung 1.1 haben die Farben, Kulturkreise und Begriffe einen nominalen Datentyp.

Ordinale Datentypen haben die gleichen Eigenschaften wie nominale Typen. Zusätzlich dazu ist die Reihenfolge allgemeingültig festgelegt. Dadurch kann als zusätzlicher Operator für Vergleiche die Richtung der Ausprägung genutzt werden. Diese ist durch eine Ordnungsrelation bestimmt. Ein Beispiel für ordinale Daten sind Temperaturwerte, die in drei Klassen „kalt“, „warm“ und „heiß“ unterteilt werden. Neben der Unterscheidung zwischen den drei Klassen kann eindeutig bestimmt werden, dass auf die Klasse „kalt“ die Klasse „warm“ und dann die Klasse „heiß“ folgt.

Quantitative Datentypen besitzen einen Wertebereich und gestatten die Durchführung von arithmetischen Operationen. Es existiert dabei eine Maßeinheit, mit der die Abstände zwischen einzelnen Ausprägungen angegeben werden können. Durch eine künstliche Einteilung in Intervalle kann dieser Typ in ordinale Datentypen übersetzt werden. Ein Beispiel ist eine Temperaturmessung, bei der Werte in Grad Celcius gespeichert werden.

2.5 Grafische Semiologie

Die grafische Semiologie ist ein von Bertin (1974) erstmals beschriebenes System zur Klassifikation von grafischen Zeichen. Es stammt ursprünglich aus der thematischen Kartografie.[16] Dies ist ein Teilgebiet der Geografie und eines der ersten Anwendungsgebiete der Informationsvisualisierung. Diese Theorie definiert ein „Grundalphabet“ an grafischen Elementen, wobei zur Informationsübertragung jedem einzelnen Zeichen eine bestimmte Bedeutung zugeordnet wird. Bertin (1974) geht dabei von einem monosemiotischen[17] System aus, das den rationalen Teil der Bilderwelt, die grafische Darstellung, eindeutig beschreibt. Da Menschen aber bestimmten grafischen Zeichen unterschiedliche Bedeutungen zuordnen, muss ein „rationales Moment“ stattfinden, bei dem sich alle an der Kommunikation Beteiligten auf Bedeutungen einigen, die bestimmte Zeichen in der grafischen Darstellung haben. Erst dann kann über die Verbindung der Zeichen untereinander diskutiert werden. Diese Verbindungen entsprechen der eigentlichen Information. Beispielsweise können durch die Position von grafischen Primitiven untereinander Aussagen über ihre eigentlichen Beziehungen getroffen werden. Dabei spielen die Gestaltungsgesetze eine wichtige Rolle. Mit der grafischen Semiologie können also Informationen in ein grafisches Zeichensystem transkribiert werden. Dieses System umfasst folgende acht Zeichen, die zur Kodierung von Daten in eine grafische Darstellung be-

[16] „Thematische Karten enthalten vorwiegend Erscheinungen oder Vorkommnisse nicht topographischer [sic] Art, welche jedoch mit der Erdoberfläche in Verbindung stehen. Es handelt sich hierbei um Dinge, die georäumliche Lage, Verbreitung oder Bewegung besitzen, sowohl um reale Dinge, als auch um Beziehungen, Funktionen, Hypothesen, geistige Vorstellungen, Möglichkeiten und Projekte.“(Gitta, 2011)

[17] „Die Betrachtung einer Zeichenverbindung setzt die Kenntnis der Bedeutung jedes einzelnen Zeichens voraus.“(Bertin, 1974, S. 3)

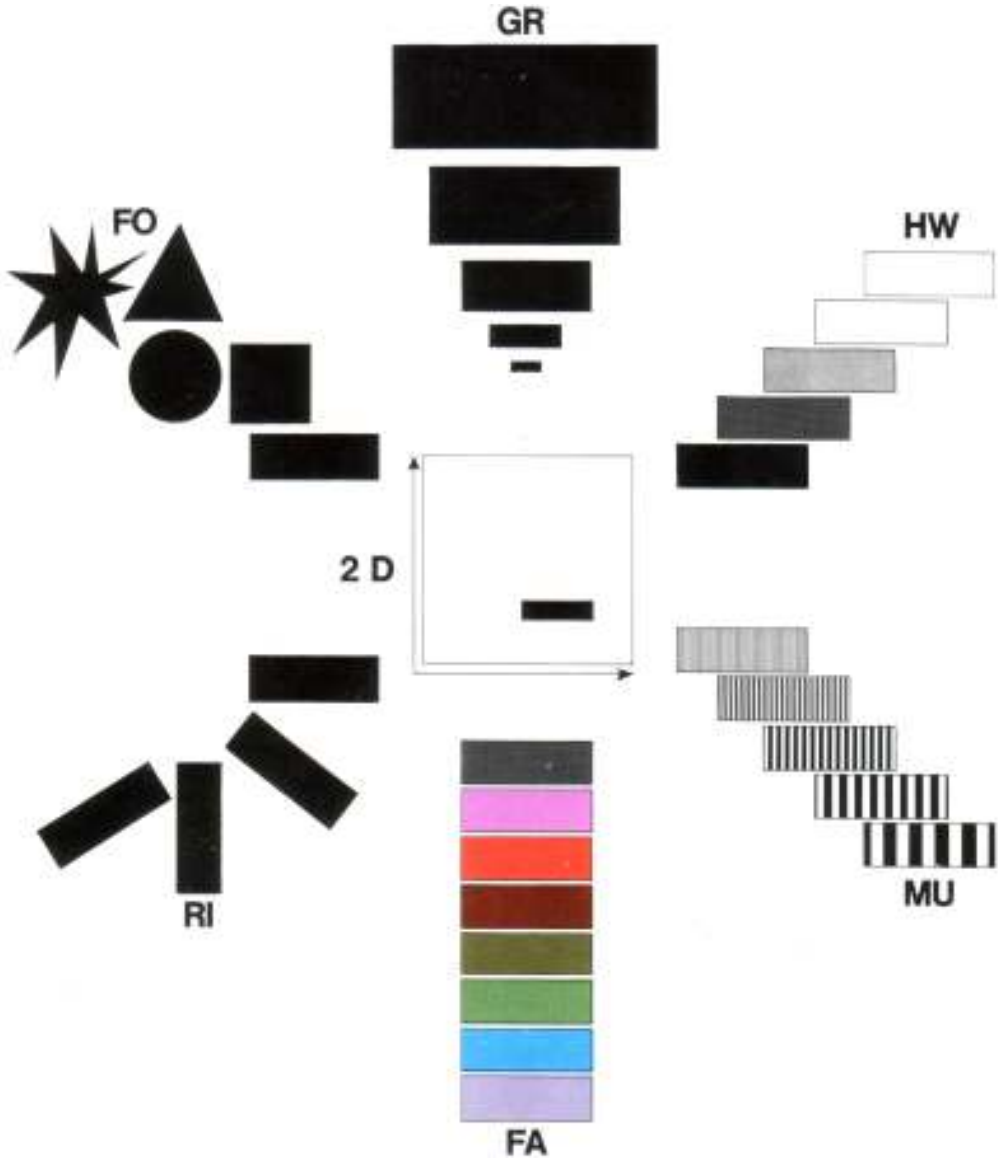


Abbildung 2.1: Visuelle Variablen aus (Bertin, 1974, S. 51) Legende: 2D = Position in der Ebene, FO = Form, GR = Größe, HW = Helligkeitswert, MU = Musterung/Textur, FA = Farbe, RI = Richtung

nutzt werden können. Sie werden visuelle Variablen genannt und beschreiben die Eigenschaften geometrischer Primitive.

- Position auf der Ebene (zwei visuelle Variablen x und y)
- Größe (Fläche oder Länge)
- Helligkeitswert
- Musterung oder Textur
- Farbe
- Richtung oder Orientierung
- Form des Elements

In Abbildung 2.1 werden diese dargestellt. Im Initialbeispiel in Abbildung 1.1 werden die visuellen Variablen der Position, Farbe und Richtung verwendet, um die Information in den kreisförmig angeordneten farbigen Rechtecken darzustellen.

Jede dieser Variablen hat spezifische Eigenschaften, wodurch sie sich unterschiedlich gut oder schlecht für spezielle Aufgaben eignen. Um für eine Aufgabe die geeignetste Variable zu bestimmen, schlägt Bertin (1974) mehrere Konstruktionsregeln vor. Grundlegend beschreiben diese eine Vereinfachung der grafischen Darstellung. Die Anzahl an visuellen Variablen, Merkmalen und Ausprägungen wird reduziert, sodass die grafische Darstellung als Ganzes mit einem Minimum an Wahrnehmungsaufwand erfasst werden kann. Dabei dürfen keine Informationen und Beziehungen verloren gehen, um Fehlinterpretationen

zu vermeiden. Weitere Regeln sind stark abhängig von der Anzahl der vorhandenen Dimensionen des Beobachtungsraumes, der Anzahl der Merkmale eines Beobachtungspunktes[18] und der gewählten, grafischen Darstellung. Da sich für die komplexen, grafischen Darstellungen einer Informationsvisualisierung meist mehrere visuelle Variablen eignen, kann eine Auswahl durch den Abgleich mit den Bearbeitungszielen[19] erfolgen. Beispielsweise eignen sich für wichtige, geordnete Variablen die Position in der Ebene, während Variablen wie Farbe, Muster und Textur als Möglichkeit genutzt werden können, um Zusatzinformationen zu codieren. Erst wenn weitere Variablen hinzugefügt werden, muss eine Nebeneinanderstellung oder Überlagerung in Betracht gezogen werden. Die Konstruktionsregeln und die visuellen Variablen werden genutzt, um die grafische Darstellung als Ziel der Informationsvisualisierung zu erstellen.

[18] Vgl. Abschnitt 2.2
[19] Vgl. Abschnitt 3.3

3 Informationsvisualisierung

3.1 Definition

Die Visualisierung ist im Allgemeinen als eine „rechnergestützte, visuelle Präsentation von Daten, Informationen und Wissen in einer für den Menschen adäquaten und für die jeweilige Anwendung in diesem Kontext sinnvollen Form zu verstehen.“[1] Sie nutzt das Vermögen des menschlichen Gehirns durch visuelle Reize große Mengen an Informationen intuitiv erfassen und verarbeiten zu können. Dabei werden die natürlichen Fähigkeiten des Menschen, wie beispielsweise die Muster- und Farberkennung, genutzt, um Entscheidungen über einen Sachverhalt ohne besondere Vorkenntnisse treffen zu können. Zum Beispiel kann jeder Mensch zwischen zwei Objekten entscheiden, welches das größere oder hellere Objekt ist, solange die visuellen Unterschiede ausreichend groß sind.[2]

Die wissenschaftlich-technische Visualisierung ist eine Sonderform der Visualisierung. Sie arbeitet ausschließlich mit Ausgangsdaten, die einen natur- oder

[1] (Schumann und Müller, 2000, S. 3)

[2] Eine Ausnahme bilden Menschen mit einer Sehbehinderung. Um eine Visualisierung für diese Menschen nutzbar zu machen, gelten besondere Anforderungen, die nicht Teil dieser Arbeit sind.

ingeneurswissenschaftlichen Bezugsrahmen und somit einen konkreten Ortsbezug haben. Diese Form der Visualiserung wird Datenvisualisierung genannt. Dabei wird von einem, durch die Informationstheorie geprägten, quantitativen Informationsbegriff ausgegangen.[3]

Für die Visualisierung von Informationen sind weiterführende, qualitative Fragen von Bedeutung:

- Was macht eine Information in einem gegebenen Kontext wichtig oder unwichtig?
- Wie kann das Verständnis von großen Informationsmengen für Menschen vereinfacht werden?
- Wie können Informationen visuell aufbereitet werden, um eine effizientere Problemlösung zu ermöglichen?

Diese Fragen, die aufgrund eines wachsenden Spektrums an abstrakten Daten immer wichtiger werden, versucht die Informationsvisualisierung zu beantworten. Preim und Dachselt (2010) definieren Informationsvisualisierung folgendermaßen:

> „Informationsvisualisierung beschäftigt sich mit der Visualisierung vorrangig abstrakter Daten, wie Multiparameterdaten (z.B. Medienobjekte mit verschiedenen Attributen), Hierarchien, Netzwerken, Text oder Softwaresystemen, die sich alle auch über die Zeit verändern können.“[4]

[3] Vgl. Abschnitt 2.1
[4] (Preim und Dachselt, 2010, S. 434)

Das Einsatzgebiet der Informationsvisualisierung ist im Gegensatz zur Datenvisualisierung nicht auf wissenschaftliche Bereiche beschränkt, sondern kann beispielsweise als alternative Suchmethode in Multimedia-Datenbanken,[5] als Wissensvermittlung in kulturellen Bereichen[6] und zur Visualisierung von sozialen Netzwerken und Fahrplänen öffentlicher Verkehrsmittel verwendet werden. Insbesondere steht die Suche nach Beziehungen zwischen Datenobjekten im Vordergrund, da Informationsräume größer, komplexer und vernetzter werden. Im Gegensatz zu naturwissenschaftlichen Daten existiert dabei nicht immer ein räumlicher Bezug. Eine Aufgabe der Informationsvisualiserung ist es, diesen Ortsbezug beim Betrachter wieder herzustellen. Dieser Ortsbezug wird bei abstrakten Daten künstlich erzeugt, da Menschen es gewöhnt sind, in dreidimensionalen Räumen zu interagieren. Somit können bewusste und unbewusste Verhaltensweisen aus der realen Welt genutzt werden, um den kognitiven Aufwand, der für das Verständnis der Information nötig ist, zu minimieren.

Durch die Verschiebung der Daten aus dem wissenschaftlichen Kontext in die Alltagskultur ändert sich die Zielgruppe der Visualisierung. Durch das Internet wird der Personenkreis stark vergrößert, der ein Interesse und Nutzen an Informationsvisualisierungen hat. Es enstehen weitere Anforderungen, da dieser Gruppe meist ein mathematischer, natur- oder ingenieurswissenschaftlicher Hintergrund fehlt. So muss die Informationsvisualisierung vom Benutzer schnell und einfach erfasst und ohne besondere Vorkenntnisse verstanden werden können. Außerdem muss sie eine „angemessene und grafische Qualität haben, um über den reinen Nutzwert hinaus auch Qualitäten in Bezug auf

[5] Vgl. „Visual Data Mining“ in (Keim, 2002)
[6] Vgl. Abbildung 1.1

Nutzungsfreude und Unterhaltungswert [zu] besitzen.“[7] Es ist schwierig solche Darstellungsformen mit aktueller Standardsoftware zu erstellen.

Die Grenzen zwischen den einzelnen Disziplinen der Visualisierung sind fließend. Die Informationsvisualisierung kann auch wissenschaftlich betrieben werden, da sich Methoden und Verfahren stark ähneln. Das Arbeitsfeld der Informationsvisualisierung, das Thema dieser Arbeit ist, geht von einem Spezialfall der Datenvisualisierung aus. Dieser erweitert die Anforderungen an Datenvisualisierung mit Betrachtungen über die Zielgruppe, die Repräsentation und das Medium. Dabei kommen digitale Techniken zur Erhebung, Speicherung, Weiterverarbeitung und Ausgabe zum Einsatz.

3.2 Verwandte Arbeitsfelder

Oft nutzt die Informationsvisualisierung Teilgebiete verwandter Arbeitsfelder oder sie wird für die Problemlösung in anderen Bereichen verwendet. Eine klare Abgrenzung der Bereiche ist daher schwierig. Nachfolgend werden Arbeitsgebiete beschrieben, die häufig mit der Informationsvisualisierung in Verbindung gebracht oder verwechselt werden. Die nachfolgende Liste zeigt eine Auswahl und erhebt keinen Anspruch auf Vollständigkeit.

Computergrafik Die Computergrafik ist ein Teil der Informatik und beschreibt die Ausgabe von zwei- und dreidimensionalen Objekten als Raster oder Vektorgrafik. Die Informationsvisualisierung benutzt Techniken der Computergrafik, ist aber kein Bestandteil dieses Gebietes.

[7] (Preim und Dachselt, 2010, S. 438)

Statistik Die Statistik ist ein Verfahren für die „hypothesengeleitete Auswertung von numerischen (quantitativen) Daten.“[8] Durch die Verwendung quantitativer Daten wird eine Verbindung zwischen Empirie und Theorie geschaffen. Neben der Mathematik ist die Visualisierung das wichtigste Werkzeug der Statistik. So wird der Visualisierungsprozess aus Abschnitt 3.4 besonders im Bereich der deskriptiven Statistik benutzt, um Erkenntnisse zu gewinnen und darzustellen. Die Informationsvisualisierung als eigenständige Disziplin benutzt Techniken der Statistik, um Daten aufzubereiten.

Präsentations- und Prozessvisualisierung Häufig werden Präsentations- und Prozessvisualisierungen als Informationsvisualisierung bezeichnet, sind aber eigenständige Gebiete. In ihnen geht es um das Sichtbarmachen von Informationen. Diese Informationen basieren aber nicht auf großen Datenmengen und können ohne Softwareunterstützung dargestellt werden. Bei der Visualisierung von Prozessen muss die Zeit als besonderer Faktor berücksichtigt werden.

Data-Mining ist der computergestützte automatisierte Versuch, Datenmengen zu ordnen und mithilfe von Algorithmen Erkenntnisse zu erhalten. Im Gegensatz dazu versucht die Informationsvisualisierung eine geeignete grafische Repräsentation bereitzustellen, die dem Nutzer die Möglichkeit gibt, diese Erkenntnisse selbst zu finden. Für die Datengewinnung ist die Informationsvisualisierung auf Techniken des Data-Mining angewie-

[8] (Schäfer, 2010, S. 23)

sen. Es gibt Bestrebungen beide Gebiete im „Visual Data Mining“[9] zu verbinden, um Synergien zu bilden.

Interface- und Interactiondesign Diese Arbeitsfelder beschreiben die Gestaltung von Benutzeroberflächen. Damit arbeiten sie, wie die Visualisierung, an der Schnittstelle zwischen Mensch und Maschine. Interfacedesign benutzt häufig Techniken der Visualisierung, um grafische Konzepte zu realisieren, da die Informationen primär visuell übertragen werden.

Computational Design Dieser von Fry (2004) eingeführte Begriff beschreibt die Vereinigung vieler Disziplinen zu einem neuen interdisziplinären Arbeitsfeld. Der Ansatz des Computational Designs ist es, die Informationsvisualisierung in einem größeren Kontext zu betrachten und dabei die technischen Bereiche der Informatik mit den künstlerischen Bereichen des Designs zu verbinden. Der Prozess des Computational Designs wird in Abschnitt 3.4 genutzt, um den Arbeitsprozess der Informationsvisualisierung zu beschreiben.

3.3 Bearbeitungsziele

Mit der Informationsvisualisierung wird die Analyse einer gegebenen Datenmenge unterstützt. Es können Ergebnisse effizient präsentiert werden. Dadurch wird die Kommunikation vereinfacht, um einen Erkenntnisgewinn bei allen Beteiligten zu schaffen. Die Visualisierung kann nach Schumann und Müller (2000) für die drei folgenden Stufen der Analyse eingesetzt werden:

[9] Vgl. (Keim, 2002)

Explorative Analyse Die explorative Analyse beschreibt die Eigenschaften von Daten als Ausgangspunkt der Forschung. Durch eine interaktive und ungerichtete Suche in den Daten kann die Hypothesengenerierung mithilfe der Informationsvisualisierung unterstützt werden. Dazu muss eine geeignete weit gefasste und nicht beschränkte Darstellung gefunden werden. Diese Informationsvisualisierungen können Ausgangspunkt für weitere Analysen sein. Das Initialbeispiel in Abbildung 1.1 kann genutzt werden, um eine weiterführende Forschungsfrage zu entwickeln: Warum wird die Farbe Rot überdurchschnittlich oft mit positiven Begriffen assoziiert?

Konfirmative Analyse Es existieren bereits eine oder mehrere Hypothesen, die mithilfe der Informationsvisualisierung überprüft werden sollen. Dabei wird eine Datenmenge hinsichtlich eines speziellen Aspekts untersucht und ausgewertet.

Präsentation Die Ergebnisse aus der Analyse der Daten werden mithilfe der Informationsvisualisierung dargestellt, um sie für Dritte verständlich zu machen und mit ihnen über den Sachverhalt kommunizieren zu können. Dabei ist eine ansprechende und ästhetische Darstellung wichtig, denn ein ansprechendes Erscheinungsbild verstärkt den Willen, sich mit einer Darstellung auseinanderzusetzen.

Um eine adäquate Informationsvisualisierung klassifizieren zu können, nennen Schumann und Müller (2000) drei Eigenschaften: Effektivität, Expressivität und Angemessenheit. Diese Eigenschaften müssen bei der Konzeption einer Informationsvisualisierungssoftware beachtet werden.

Effektivität Die Darstellung muss zu den visuellen Fähigkeiten des Betrachters und den Eigenschaften des Ausgabegerätes passen.

Expressivität Die Informationsvisualisierung soll nur die in den Daten enthaltenen Informationen zeigen. Diese sollen dabei möglichst unverfälscht wiedergeben werden, um falsche Schlussfolgerungen bei dem Betrachter zu vermeiden.

Angemessenheit Ressourcen und Rechenaufwand und damit die Kosten einer Informationsvisualisierung sollen den Anforderungen entsprechen.

Bertin (1974) beschreibt dagegen nur den Begriff der Prägnanz als Qualitätsmaß. Prägnanz wird definiert:

> „Wenn eine Konstruktion zur Beantwortung einer gestellten Frage unter sonst gleichen Voraussetzungen eine kürzere Betrachtungszeit erfordert als eine andere Konstruktion, so bezeichne man diese als prägnanter in Bezug auf die gestellte Frage.“[10]

Darüber hinaus fällt es schwer, allgemeingültige Aussagen zu treffen, wie ein komplexer Sachverhalt und dessen Informationen am effizientesten zu visualisieren sind. Die Informationsvisualisierung ist stark anwendungsabhängig und soll immer den Anforderungen der Daten, der Zielgruppe, des Bearbeitungsziels, der Repräsentation und des Mediums entsprechen. Eine rein automatisierte Informationsvisualisierung kann nur sehr schwer alle diese Anforderungen berücksichtigen. Daher sollte der Benutzer in allen Stufen des Visualisierungsprozesses eingreifen und gegebenenfalls Anpassungen vornehmen können. Eine Software muss dazu geeignete Mittel zur Verfügung stellen.

[10] (Bertin, 1974, S. 17)

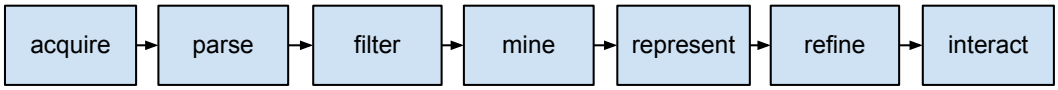


Abbildung 3.1: Der Prozess des Computational Designs aus (Fry, 2004, S. 13)

3.4 Visualisierungsprozess

Der Prozess der Visualisierung wird Visualisierungspipeline genannt.[11] Diese Pipeline beschreibt den Ablauf aller Schritte, die notwendig sind, um aus Daten eine grafische Darstellung zu erstellen. In der Literatur gibt es keine einheitliche Beschreibung der Visualisierungspipeline, jedoch ähneln sich die Darstellungen stark. Im Folgenden soll der Ablauf der Informationsvisualisierung beschrieben werden. Von Fry (2004) wird der Prozess in einen größeren Kontext eingeordnet. Dadurch eignet sich diese Visualisierungspipeline besonders gut, um einen Überblick über alle Arbeitsschritte zu erhalten. In Abbildung 3.1 wird der Prozess schematisch dargestellt.

Als Erstes werden Daten aus einer Datei oder einem Netzwerk beschafft (acquire). Danach werden Informationen aus der Datenquelle extrahiert (parse). Wichtige Informationen können beispielsweise Extremwerte oder Wertebereiche und andere Bereiche speziellen Interesses sein. Diese Bereiche werden herausgetrennt (filter), um auf ihnen Methoden der Datenanalyse anwenden zu können, die für die weitere Verarbeitung notwendig sind (mine). Im nächsten Schritt werden dann geeignete visuelle Variablen[12] gefunden, um die Daten

[11] Vgl. (Schumann und Müller, 2000, S. 15)
[12] Vgl. Abschnitt 2.5

darzustellen (represent).[13] Dieser Schritt wird als *Mapping* bezeichnet und ist eine Daten-zu-Geometrie-Abbildung[14], bei der die abstrakten Daten auf visuelle Variablen übertragen werden. Die grafische Darstellung wird dann iterativ durch die visuelle Analyse des Nutzers verbessert, damit sie einfacher verständlich und ästhetisch ansprechend wird und die Bearbeitungsziele erfüllt (refine). Als letzter Schritt werden Interaktionsmöglichkeiten ergänzt, um dem Nutzer die Kontrolle über die Ausgabe, aber auch über vorher stattfindende Stufen des Prozesses, geben zu können (interact).

Der hier beschriebene Ablauf ist allen Visualisierungsprozessen immanent, muss aber nicht zwangsläufig in der gegebenen Reihenfolge auftreten. Es gibt unterschiedliche Möglichkeiten, in die Visualisierungspipeline einzugreifen und zwischen den einzelnen Stufen zu wechseln.[15] Das Ergebnis dieser Verfahren ist jedoch immer die grafische Darstellung.

3.5 Grafische Darstellung

3.5.1 Begriff

Die grafische Darstellung ist das Ergebnis der Informationsvisualisierung. Sie besteht aus den in Abschnitt 2.5 genannten visuellen Variablen. Als allgemeinste Form der grafischen Darstellung beschreiben Schumann und Müller (2000) das Diagramm. Es „ist eine graphische [sic] Repräsentation von Informa-

[13] Schumann und Müller (2000) erweitern diesen Punkt durch das Rendering, welches die eigentliche Rasterung und Darstellung auf dem Ausgabemedium durchführt.

[14] Vgl. (Schumann und Müller, 2000, S. 16)

[15] In (Schumann und Müller, 2000) Abschnitt 2.2 und 2.3 werden unterschiedliche Verfahren beschrieben.

tionen mit Hilfe graphischer Elemente.“[16] Bertin (1974) spricht dagegen vom Begriff der „grafischen Darstellung“ und grenzt diesen von der bildhaften Darstellung[17] ab, die nicht monosemiotisch[18] ist. Diese Klassifikation wird weiter in Diagramme, Netze und Karten unterteilt. Schumann und Müller (2000) weisen außerdem auf eine Doppeldeutigkeit des deutschen Begriffes „Diagramm“ hin, denn in der englischen Sprache gibt es „diagram“ und „chart“. Dort beschreibt „diagram“ die Darstellung nicht-quantitativer Beziehungen, während „chart“ für die allgemeine Form der grafischen Darstellung steht.[19] In dieser Arbeit wird der Begriff der *grafischen Darstellung* verwendet und schließt damit alle möglichen Darstellungsformen der Informationsvisualisierung ein.

3.5.2 Darstellungsformen

Die grafische Darstellung hat verschiedene Darstellungsformen, die für unterschiedliche Anwendungsfälle konzipiert wurden. Sie sind alle stark anwendungsabhängig. Es müssen viele Anforderungen beachtet werden, um die effizienteste Technik der grafischen Darstellung zu finden.[20] Die folgende Auflistung von Darstellungsformen orientiert sich an den Darstellungsformen für Multiparameterdaten von Schumann und Müller (2000) und wird ergänzt durch Preim und Dachselt (2010) und eigene Beobachtungen. Aufgrund der Vielzahl an unterschiedlichen Darstellungsformen kann die Tabelle 3.1 nur einen beispielhaf-

[16] (Schumann und Müller, 2000, S. 126)
[17] In Bertin (1974) werden Fotos oder künstlerische Bilder explizit ausgeschlossen.
[18] Vgl. Abschnitt 2.5
[19] (Schumann und Müller, 2000, S. 126)
[20] Vgl. Abschnitt 3.3

Technik	**exemplarische Darstellungsformen**
einfach	Scatterplots (1), Histogramme (2), Kreisdiagramme (3), Isolinien (4)
multiaxial	Kiviatgraph (5), parallele Koordinaten (6)
ikonenbasiert	Stick-Figure-Ikonen (7), Chernoff Ikonen (8), Data Jacks (9)
pixelbasiert (10)	Zwei-Schritte-Technik, Recursive-Pattern-Technik
hierarchisch	dimensional Stacking (11), World-within-Worlds (12), Cone-Trees (13), Baumdiagramm (15)
netzwerkbasiert	Venn-Diagramm (14), Treemaps (17), Bubble Tree (16), ArcTree (20)
geschichtet	Icicle Plots (19), Sunburst (18)
hybrid	Mischtechniken bestehend aus anderen Techniken

Tabelle 3.1: Techniken der Darstellung

ten Überblick geben. Die Nummern hinter den einzelnen Darstellungsformen verweisen auf die Darstellungen in Abbildung 3.2.[21]

Für diese Arbeit sind vor allem die hybriden Techniken interessant, da diese versuchen, einzelne Vorteile verschiedener Darstellungsformen zu verbinden, um neue Sichtweisen zu erforschen. Damit solche Informationsvisualisierungstechniken gefunden werden können, muss eine Konzeption für eine Softwarelösung Einschränkungen bei der grafischen Darstellung vermeiden.

[21] Ein Beispiel für die Recursive-Pattern-Technik findet sich im Anhang in Abschnitt 8.3, da pixelbasierte Techniken nur sehr schwer schematisch dargestellt werden können.

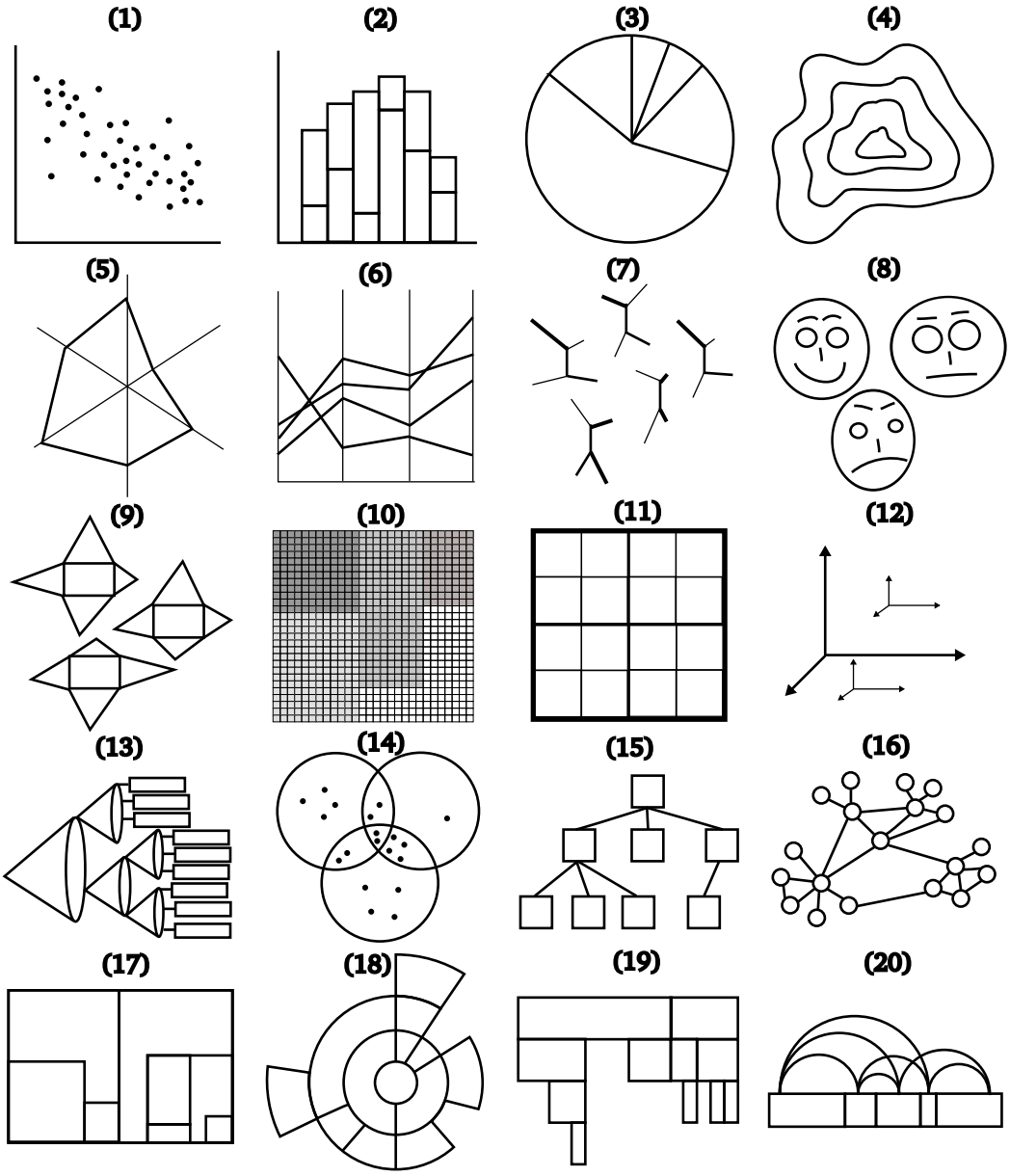


Abbildung 3.2: Verschiedene Darstellungsformen, eigene Darstellung

4 Konzeption

4.1 Allgemeine Betrachtungen

Die Übertragung von abstrakten Daten auf visuelle Variablen ist das grundlegende Prinzip der Informationsvisualisierung. In der Fassung von Fry (2004) wird diese Transkription in den Stufen *represent* und *refine* durchgeführt. Die Software muss also dem Benutzer die Möglichkeit geben, diese Arbeitsschritte abwechselnd durchführen zu können. In Abbildung 4.1 ist dieser weitere iterative Schritt abgebildet. Ist der Benutzer mit der grafischen Darstellung nicht zufrieden, so kann er Parameter anpassen. Dieser Ablauf kann beliebig oft wiederholt werden bis ein zufriedenstellendes Ergebnis vorliegt. Die hohe Anzahl der Beobachtungspunkte eines Datensatzes macht eine vom Nutzer manuell

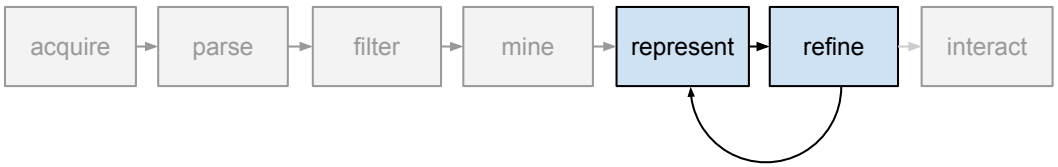


Abbildung 4.1: Einordnung in den Prozess des Computational Designs nach (Fry, 2004)

durchgeführte Transkription sehr aufwendig und zeitintensiv. Eine Unterstützung des Benutzers durch eine Software ist daher sinnvoll. Dabei darf der Nutzer in seinem kreativen Gestaltungsprozess nicht eingeschränkt werden.

Wie in Kapitel 2 und 3 gezeigt wurde, gibt es eine Vielzahl an Datenklassen und grafischen Darstellungsformen. Weiterhin entstehen neue hybride Formen durch Mashups.[1] Die Software muss daher möglichst universell sein, um flexibel viele unterschiedliche Anwendungsfälle bearbeiten zu können. Dabei dürfen die Bearbeitungsziele aus Abschnitt 3.3 nicht vernachlässigt werden.

4.2 Anforderungen

Für die zu entwickelnde Software werden Anforderungen festgelegt, die im Folgenden beschrieben werden. Neben speziellen Anforderungen, die aus dem Bereich der Informationsvisualisierung stammen, werden noch weitere Anforderungen an Benutzerführung und Performanz gestellt.

Einfache Benutzerinteraktion Damit technisch unerfahrene Nutzer schnell und einfach Ergebnisse erzielen können, kann die Software mit visueller Programmierung gesteuert werden. Für die Bedienung der Software mithilfe der visuellen Programmierung, die in Abschnitt 4.3 genauer beschrieben wird, reicht ein grundlegendes mathematisches Verständnis des Nutzers aus. Weiterhin veranschaulicht sie den eigentlichen Anwendungsbereich der Software.

[1] „Mashup (von englisch „to mash“ für vermischen) bezeichnet die Erstellung neuer Medieninhalte durch die nahtlose (Re-)Kombination bereits bestehender Inhalte.“(Mashup, 2011)

Flexible Visualisierungsmöglichkeiten Um möglichst viele Darstellungsformen abbilden zu können, muss die Möglichkeit bestehen, alle visuellen Variablen[2] und damit alle Darstellungsformen[3] reproduzieren zu können. Der Nutzer darf dabei in seiner Arbeit nicht durch vorgefertigte Darstellungsformen eingeschränkt werden.

Flexible Ein- und Ausgabe Zur automatisierten Dateneinspeisung in die Software sollen mehrere Möglichkeiten bestehen, um in unterschiedliche Workflows eingebunden werden zu können. Neben Dateiformaten wie XML, JSON und CSV ist eine Netzwerk- und Datenbankanbindung sinnvoll. Als Ausgabeformate können gängige Bildformate verwendet werden.

Performanz Der Benutzer kann alle Verbindungen und Parameter zwischen Datenobjekten und ihrer visuellen Repräsentation direkt beeinflussen. Für eine direkte visuelle Analyse durch den Nutzer muss die Software die Möglichkeit bieten, sofort nach Interaktion des Nutzers die Veränderungen darzustellen. Dafür wird eine ausreichende Performanz benötigt, um störende Lade- und Berechnungszeiten zu vermeiden.

Erweiterbarkeit Durch unterschiedliche Anforderungen in den verschiedenen Arbeitsbereichen ist es nötig, die Software durch den Benutzer erweitern zu können. Dafür soll eine Skriptsprache implementiert werden, die es erlaubt, zur Laufzeit neue Funktionen hinzuzufügen. Außerdem besteht, durch die Kapselung aller Funktionen, die Möglichkeit, dass der Benutzer eigene Module programmiert.

[2] Vgl. Abschnitt 2.5
[3] Vgl. Abschnitt 3.5.2

Plattformunabhängigkeit Die Software soll auf den drei großen Plattformen Linux, Mac OS X und Microsoft Windows lauffähig sein. Durch die Entwicklung auf Basis von openframeworks[4] wird diese Plattformunabhängigkeit gewährleistet.

Frei und offen Die Software wird unter einer freien Lizenz entwickelt und veröffentlicht, um Erweiterungen von Benutzern möglich zu machen. Durch die Wahl der Lizenz kann die Entwicklung durch alle Interessenten vorangetrieben werden. Außerdem lässt die Lizenz eine kommerzielle Erweiterung und Nutzung zu. Für Anwender ist die Software kostenlos verfügbar.

4.3 Visuelle Programmierung

Um dem breiten Spektrum an Anforderungen gerecht zu werden, wird die Software mit einer Form der visuellen Programmierung gesteuert. Henning und Vogelsang (2007) definieren visuelle Programmierung wie folgt:

> „Es handelt sich hierbei um ein in integrierten Entwicklungsumgebungen mit grafischer Benutzeroberfläche verwendetes Hilfsmittel, bei welchem grafisch dargestellte Programmblöcke durch gerichtete und qualifizierte Linien miteinander verlinkt werden.“[5]

Im Gegensatz zur textuellen Programmierung, bei der Buchstaben, Zeichen und Wörter verwendet werden, um einen Quellcode zu schreiben, wird die

[4] Vgl. Abschnitt 5.1
[5] (Henning und Vogelsang, 2007, S. 56)

```
a = 100;
b = 20;

c = a + b;
d = a - b;
e = a * b;
f = a / b;

echo c;
echo d;
echo e;
echo f;
```

Abbildung 4.2: Pseudocode für eine textuelle Programmierung aller Grundrechenarten.

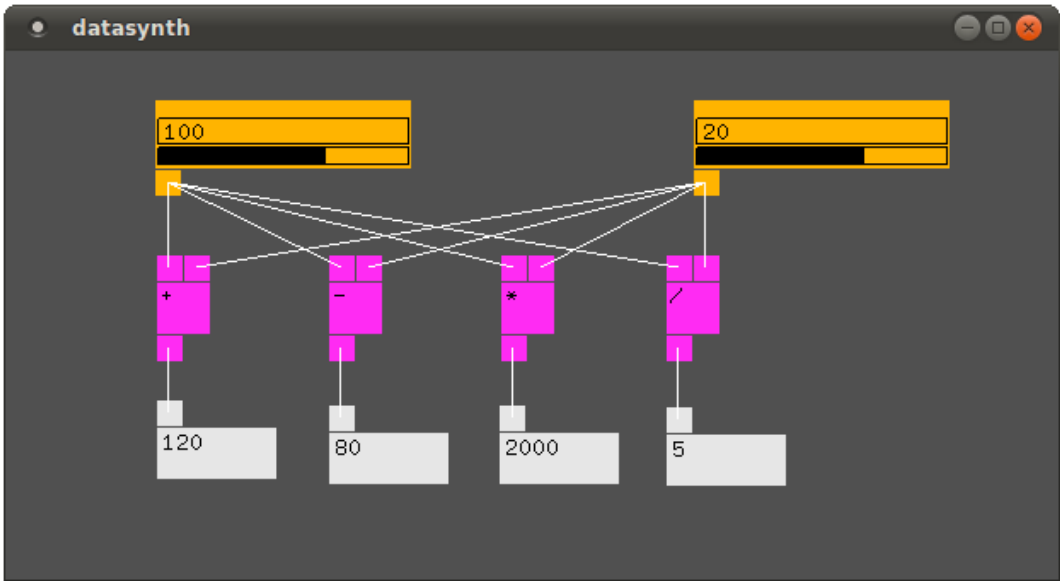


Abbildung 4.3: Die Grundrechenarten umgesetzt mit visueller Programmierung.

Programmlogik bei ber visuellen Programmierung durch grafische Elemente erstellt. Statt vorgefertigte Funktionsnamen und Aufrufe zu schreiben, werden einzelne grafische Elemente mit unterschiedlichen Aufgaben erzeugt und diese untereinander verbunden. Die visuelle Programmierung wird fast ausschließlich mit der Maus durchgeführt. Als Beispiel wird in Abbildung 4.2 eine textuelle Programmierung mit Pseudocode gezeigt, die zwei Variablen mit den vier Grundrechenarten verbindet. In Abbildung 4.3 wird die gleiche Berechnung in visueller Programmierung umgesetzt.

Für die Software werden einzelne Klassen definiert, die in ihrer Summe das Konzept der visuellen Programmierung bilden. In der folgenden Auflistung werden die einzelnen Elemente und ihre Funktionsweise beschrieben.

Node Ein Node ist ein Knoten in dem Funktionalität gekapselt ist. Vergleichbar ist dieses Element mit einer Funktion oder Methode in der textuellen Programmierung. Ein Node kann also beliebig viele Parameter zur Eingabe haben. Als Ausgabe sind mehrere unterschiedliche Werte möglich. Ein Node kann zwischen der Eingabe und der Ausgabe unterschiedlichste Berechnungen durchführen. Diese können unter anderem arithmetischer oder logischer Natur sein. Ein Node kann beispielsweise Zahlen miteinander multiplizieren oder bei einem bestimmten Grenzwert von true nach false umschalten. Alle Berechnungen in einem Node werden unabhängig von allen anderen vorhandenen Nodes durchgeführt.

Pin Ein Pin beschreibt die unterschiedlichen Ein- und Ausgabewerte eines Node. Es wird dabei zwischen Input- und Output-Pins unterschieden. Über die unterschiedlichen Input-Pins können verschiedene Werte übergeben werden, die für eine Berechnung in dem Node notwendig sind. Diese wer-

den von dem Node verarbeitet und die Ergebnisse dieser Berechnung über die Output-Pins ausgegeben. Vergleichbar sind die Input-Pins mit den Parametern einer Funktion in der textuellen Programmierung. Die Output-Pins können als Rückgabewerte einer Funktion verstanden werden.

Connection Eine Connection ist die Verbindung zwischen zwei Pins. Dabei wird vom Benutzer ein Output-Pin einer Node mit dem Input-Pin einer anderen Node verbunden. Nur durch diese Verbindung können Nodes miteinander kommunizieren und Werte austauschen. Eine Connection ist mit einer Zuweisung oder gegenseitigen Funktionsaufrufen in der textuellen Programmierung vergleichbar.

Spread Ein Spread ist eine Struktur, die in der Software als universeller Datentyp funktioniert.[6] Eingabe- und Ausgabewerte von Nodes können nur Spreads sein. Ein Spread ist eine eindimensionale Liste von Werten. Diese Werte können unterschiedliche, primtive Datentypen, wie beispielsweise Gleitkommazahlen oder Zeichenketten, beinhalten. Im einfachsten Fall hat die Liste eines Spreads nur ein einziges Element und ist demzufolge eine einfache Variable. In komplexeren Anwendungen kann ein Spread eine große Anzahl an Elementen enthalten, die beispielsweise aus einer Datentabelle stammen. Somit kann der Benutzer eine Reihe von Zahlen mit einer Konstante multiplizieren, ohne die Anzahl der einzelnen Werte eines Spreads beachten zu müssen. Die Software ist in der Lage, Spreads mit einer unterschiedlichen Anzahl von Elementen zu verbinden.

[6] Der Begriff Spread wurde aus der Software „vvvv“ übernommen. Vgl. Abschnitt 4.5

Der Einsatz von visueller Programmierung als Bedienkonzept hat mehrere Vorteile. Das Verbinden von Nodes, die als Datenquelle funktionieren, mit Nodes, die die visuellen Variablen repräsentieren, spiegelt das grundlegende Prinzip der Informationsvisualisierung in der Benutzeroberfläche wider. Für den Benutzer ist es dadurch verständlicher welche Aufgabe die Software hat und wie mit ihr Probleme gelöst werden können. Mit der gewählten Benutzeroberfläche sind keine Programmierkenntnisse erforderlich, um eine Informationsvisualisierung zu erstellen. Ein durchschnittliches mathematisches Verständnis des Benutzers reicht für einfache Aufgaben aus. Das macht ein iteratives Experimentieren möglich. Der Benutzer kann die Software einsetzen, um eine Idee für eine Informationsvisualisierung unkompliziert zu skizzieren. Die komplexen internen Abläufe liegen dabei hinter einem intuitiven und einfachen Interface verborgen. Dadurch können „context switches“ [7] vermieden werden. Diese treten auf, wenn der Benutzer zwischen unterschiedlichen Programmansichten wechseln muss. Stattdessen wird der Bereich für die Benutzerinteraktion und die Ausgabe immer gleichzeitig angezeigt. Der Nutzer erkennt dadurch sofort, welchen Einfluss seine Benutzereingaben haben. Er kann diese vergleichen und anpassen um schnell zum gewünschten Ergebnis zu kommen. In Abbildung 4.4 wird dieses Prinzip an einem Beispiel dargestellt.

[7] Vgl. (Tufte, 1990, S. 50)

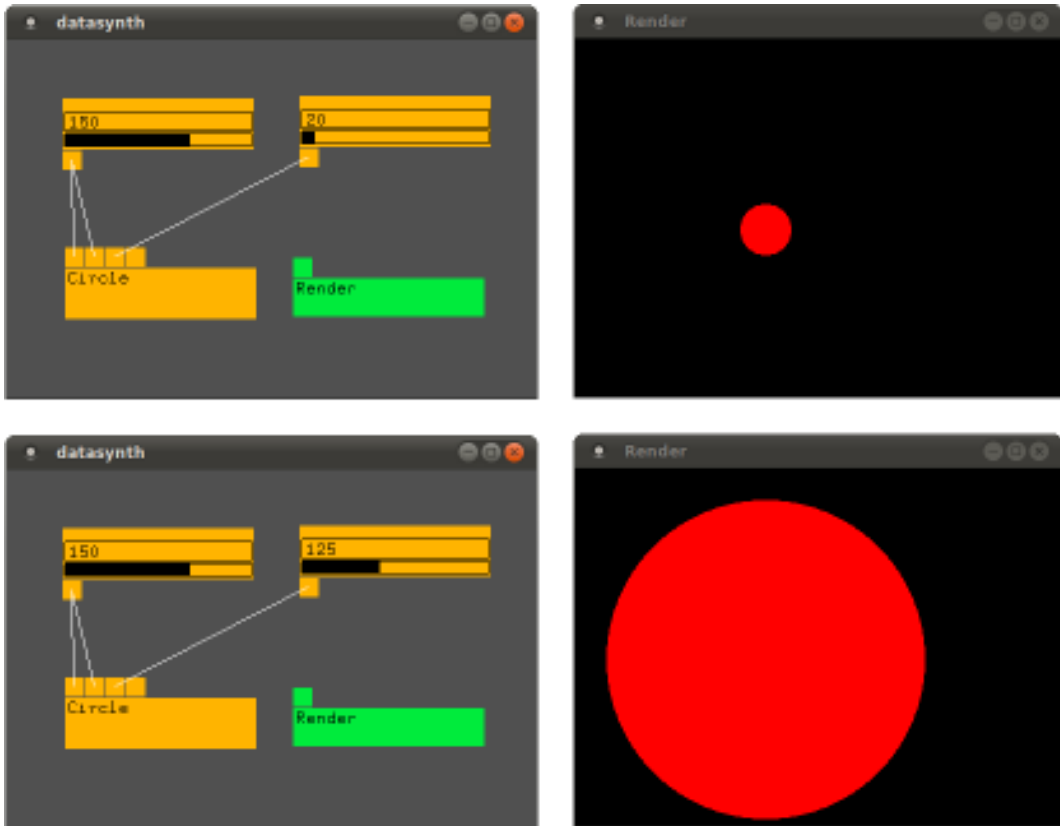


Abbildung 4.4: Ein Beispiel für eine Benutzeroberfläche der visuellen Programmierung. Zwischen der oberen Reihe und der unteren Reihe von Fenstern hat eine Benutzerinteraktion stattgefunden. Im oberen rechten Fenster wird ein Kreis gezeichnet. Dieser hat einen Radius von 20 Pixel, da die rechte obere Node im linken Fenster einen Wert von 20 Pixel an den Pin für den Radius des Kreises übergibt. In der unteren Reihe wurde der Wert auf 125 geändert. Dadurch verändert sich die Darstellung unmittelbar, da der Radius des Kreises mit dem Wert dieser Node verbunden ist. Die Position des Kreises ist in beiden Darstellungen bei 150 Pixel in der Höhe und Breite, da der obere linke Node den Wert 150 hat und mit zwei Verbindungen die x- und y-Position des Kreises beschreibt.

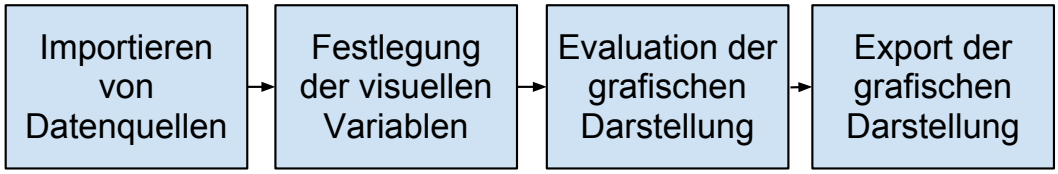


Abbildung 4.5: Schematische Prozesskette der Software

4.4 Prozesskette

In der Prozesskette wird der generische Arbeitsablauf für die Benutzung der Software beschrieben. Dieser ist unabhängig von den zu verarbeitenden Daten und den gewählten Darstellungsformen. Die Prozesskette ist in vier Arbeitsschritte unterteilt, die nacheinander abgearbeitet werden müssen, um eine grafische Darstellung zu erzeugen. Die einzelnen Schritte der Prozesskette werden in Abbildung 4.5 dargestellt und im Folgenden beschrieben.

4.4.1 Importieren von Datenquellen

Der Benutzer erzeugt einen Node, der es möglich macht, externe Datenquellen in die Software zu laden. Unterschiedliche Nodes haben die Aufgabe, Datenquellen auf dem Computer oder im Netzwerk zu importieren. Dabei werden etablierte Dateiformate wie XML, CSV, JSON und netCDF unterstützt. Wenn der Benutzer eine Datenquelle ausgewählt hat, erhält der Node automatisch Output-Pins. Diese repäsentieren die ursprüngliche Datenstruktur der Datenquelle. Die Datenquelle für das Beispiel aus Abbildung 1.1 kann eine Tabelle sein, bei der in der ersten Spalte alle Begriffe stehen und jede weitere Spalte die Farben eines jeden Kulturkreises beschreibt. Bei zehn Kulturkreisen

hat die Tabelle elf Spalten und der Node besitzt dadurch elf Output-Pins. Die Spreads, die durch die Output-Pins ausgegeben werden, können unterschiedlich weiterverarbeitet werden. Die Erzeugung mehrerer unterschiedlicher Nodes für die Einbindung von Daten macht ein *Mashup* unterschiedlicher Datenquellen möglich.

4.4.2 Festlegung der visuellen Variablen

Der Benutzer hat die Möglichkeit weitere Nodes anzulegen, welche die visuellen Variablen darstellen. Diese können unabhängig voneinander erzeugt und mit einem datengebenden Node verbunden werden. Sie erzeugen beispielsweise grafische Formen wie Kreise oder Rechtecke, die in ihrer Position, Farbe und Richtung beeinflusst werden können. Der Benutzer hat die Möglichkeit, zwischen datengebenden Nodes und Nodes, die eine grafische Ausgabe erzeugen, weitere Nodes einzufügen, um diese Verbindungen parametrisieren zu können. Das erlaubt unterschiedlichste Darstellunsgformen. Um das Beispiel aus Abbildung 1.1 zu realisieren, muss der Benutzer einen Node erzeugen, der Rechtecke zeichnet. Dieser Node hat Input-Pins um die Position in der Ebene, die Größe, die Rotation und die Farbe festlegen zu können. Der Benutzer verbindet dann die Output-Pins des datengebenden Nodes mit den Input-Pins dieser Node und erzeugt damit eine grafische Darstellung.

4.4.3 Evaluation der grafischen Darstellung

Um die aktuelle grafische Darstellung überprüfen zu können, kann der Benutzer Nodes erzeugen, die eine Vorschau ermöglichen. Ein Render-Node erzeugt

ein weiteres Fenster und zeigt die Ausgabe, die von den vorhandenen Nodes erzeugt wird, an. Bei jeder Interaktion des Benutzers, die Parameter einzelner Nodes ändert, wird automatisch und unmittelbar die aktuelle Ausgabe verändert. Der Benutzer kann visuell analysieren, welche Verbindungen noch nötig sind oder welche zusätzlichen Parameter einzelne Verbindungen noch brauchen, um eine sinnvolle grafische Darstellung zu erzeugen. Ist das Beispiel aus Abbildung 1.1 mit den vorangegangen Schritten noch nicht zufriedenstellend visualisiert, kann der Benutzer weitere Nodes und Connections hinzufügen. So kann beispielsweise eine zusätzlicher Multiplikator-Node und ein Variable-Node genutzt werden, um die Position aller Rechtecke anzupassen.

4.4.4 Export der grafischen Darstellung

Ist der Benutzer zufrieden mit der grafischen Darstellung, kann er weitere Nodes erzeugen, die diese exportieren. Zur Speicherung von grafischen Darstellungen eignen sich bekannte Bildformate wie JPEG, PNG, BMP oder TIFF. Nodes, die andere Formate für die Weiterverarbeitung unterstützen, sind denkbar. Weitere Exportmöglichkeiten sind Vektorgrafiken, 3D-Austauschformate oder interaktive HTML Seiten.

4.5 Bestehende Softwarelösungen

Durch das breite Arbeitsfeld der Informationsvisualisierung, das meist fließend in andere Bereiche übergeht, existiert eine große Anzahl an Softwarelösungen und Programmierbibliotheken. Hier sollen exemplarisch einzelne Anwendungen beschrieben werden, die verschiedene, geforderte Funktionen enthalten,

diese aber nicht in einem Softwarepaket, das speziell auf die Informationsvisualisierung zugeschnitten ist, zusammenfassen. Eine Liste aller betrachteten Programme findet sich im Anhang in Abschnitt 8.1.

vvvv ist eine datenstromorientierte Enticklungsumgebung, die universell eingesetzt werden kann. Mit ihr können unterschiedlichste, audiovisuelle Medien gekoppelt und verarbeitet werden. vvvv wird durch visuelle Programmierung gesteuert, ist aber nicht spezialisiert auf Informationsvisualisierung und nicht für alle Plattformen verfügbar.

MeVisLab ist eine Software aus dem medizinischen Bereich, die durch visuelle Programmierung bedient werden kann. MeVisLab ist eine fortgeschrittene integrierte Entwicklungsumgebung mit verschiedenen Erweiterungsmöglichkeiten, ist aber spezialisiert auf die medizinische Bildverarbeitung.

d3.js ist eine Bibliothek, die Elemente einer HTML-Seite mit Daten verbinden kann. Sie ist in Javascript geschrieben und beschränkt sich auf Webanwendungen. Der Nutzer muss Javascript programmieren, um diese Bibliothek nutzen zu können.

NodeBox 2 Beta ist ein Programm, um generative Grafiken zu erstellen. Es entspricht nahezu allen beschriebenen Anforderungen. Für komplexere Aufgaben muss aber die Programmiersprache Python erlernt werden. Die Anbindung von unterschiedlichen Datenquellen ist nicht gegeben.

5 Prototypische Realisierung

In diesem Kapitel wird die in Kapitel 4 beschriebene Konzeption prototypisch implementiert. Dieser Prototyp trägt den Namen *Datasynth*. Aufgrund der begrenzten Arbeitszeit werden nur grundlegende Funktionen umgesetzt, die notwendig sind, um das Prinzip der Software zu verdeutlichen.

5.1 Technologien

Um Datasynth zu entwickeln, werden verschiedene Technologien verwendet. Im Folgenden wird ihr Einsatz bei der Entwicklung beschrieben.

C++ ist eine objektorientierte Programmiersprache. Sie bietet eine maschinennahe Möglichkeit, effiziente Programme zu entwickeln. Datasynth nutzt durch openFrameworks und boost die Vorteile von C++. Dazu gehören eine höhere Geschwindigkeit beim Ausführen und eine einfache Einbindung anderer Bibliotheken, wie beispielsweise OpenGL für Grafikoperationen. Außerdem findet das Programmierparadigma der Objektorientierung in den einzelnen Nodes von Datasynth Anwendung.

openFrameworks ist ein in C++ geschriebenes Framework. Es soll durch seine Einfachheit und Intuitivität den kreativen Prozess und das Experi-

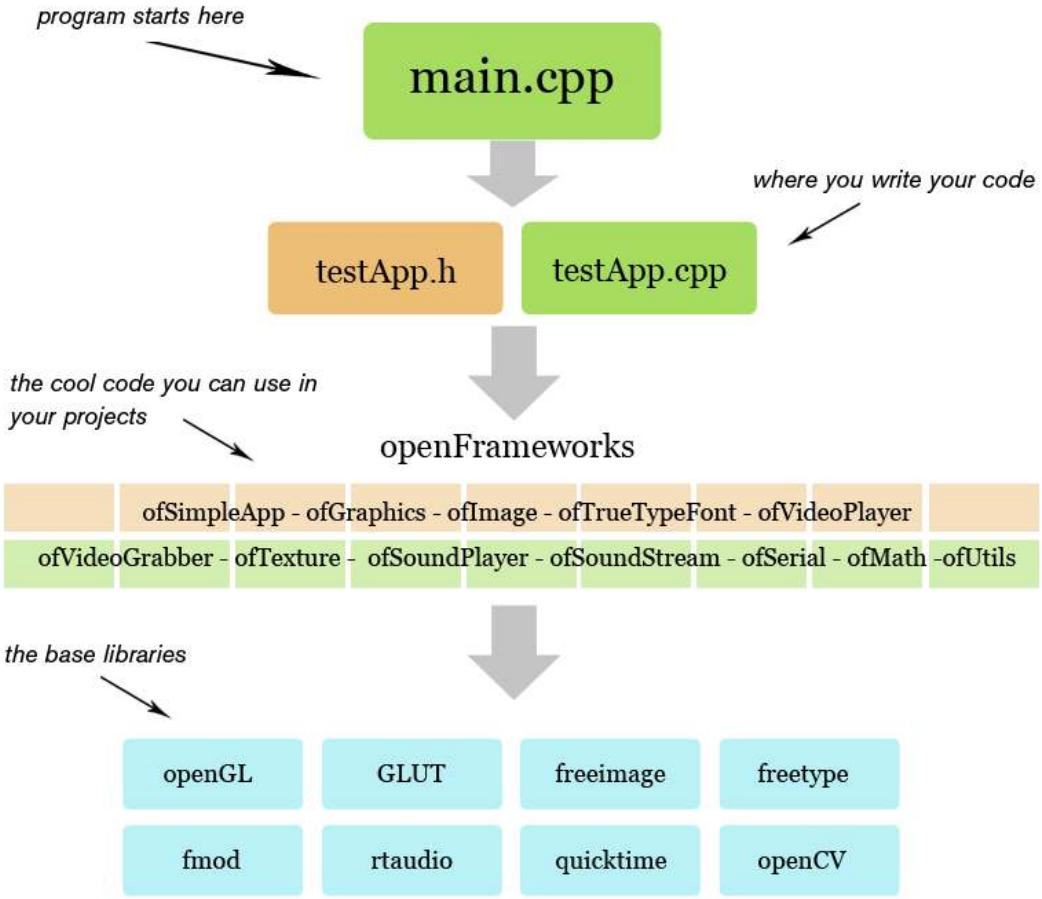


Abbildung 5.1: Funktionsweise von openFrameworks aus (openFrameworks Wiki, 2011)

mentieren beim Programmieren unterstützen. openFrameworks verbindet viele unterschiedliche Bibliotheken für die computergestützte, audiovisuelle Verarbeitung zu einer gemeinsamen, konsistenten und einfachen Programmierschnittstelle. Beispielsweise wird die Bibliothek openGL für Grafiken verwendet und die Bibliothek FreeImage für das Importieren, Bearbeiten und Exportieren von Bilddateien. openFrameworks erlaubt dem Programmierer eine schnelle und effiziente Arbeitsweise ohne lange Einarbeitungszeiten und ist daher für die Implementation von Datasynth optimal geeignet. Die Programmierschnittstelle von openFrameworks enthält wenige Klassen und Funktionen, um überschaubar zu bleiben.[1] Für spezielle Anforderungen kann auf die Befehle der zugrundeliegenden Bibliotheken zurückgegriffen werden. openFrameworks ist kompatibel zu den Plattformen Linux, Mac OS X und Microsoft Windows. Die Funktionsweise von openFrameworks wird in Abbildung 5.1 verdeutlicht. Die Programmlogik, das Benutzerinterface und alle Ein- und Ausgaben für den Benutzer wurden für Datasynth mit openFrameworks realisiert.

boost ist eine Sammlung von unterschiedlichen C++ Programmierbibliotheken. Diese Bibliotheken sind portabel einsetzbar und steigern die Produktivität bei der Entwicklung einer Software, da sie unterschiedliche Anwendungsbereiche von C++ erweitern und vereinfachen. Einzelne Bibliotheken der Sammlung wurden in den neuen C++ Standard übernommen, um C++ selbst zu verbessern. Für Datasynth werden die Biblio-

[1] Vgl. (openFrameworks Geschichte, 2011) und (openFrameworks FAQ, 2011)

theken für Zeiger, Container und generische Programmierung verwendet, um die Entwicklung zu vereinfachen.

GitHub ist eine Internetplattform für das kollaborative Entwickeln einer Software. Sie basiert auf dem Versionverwaltungstool git, welches entwickelt wurde, um Änderungen an Quellcode aufzuzeichnen und diesen zwischen mehreren Entwicklern teilen zu können. GitHub verbindet die Funktionalität von git mit einem sozialen Netzwerk, bei der verschiedene Nutzer gemeinsam an einer Software arbeiten können. Die Entwicklung von Datasynth kann online verfolgt werden.[2]

5.2 Wichtige Kernfunktionen

In diesem Kapitel werden exemplarisch wichtige Abschnitte der Software anhand ihres Quelltextes erläutert. In Abbildung 5.2 wird eine Struktur dargestellt, die die einzelnen Nodes zur Laufzeit erstellt. In dieser Struktur ist eine Liste gespeichert, die Wertepaare enthält. Ein Wertepaar besteht aus dem Namen eines Node als Zeichenkette und einer Funktion zum Anlegen dieses Node. Wenn der Benutzer einen Node erstellen will und im Menü auf einen entsprechenden Eintrag geklickt hat, wird eine Zeichenkette übergeben. Wenn die Liste der Struktur eine solche Zeichenkette enthält, wird der entsprechende Node erzeugt. So können vom Benutzer dynamisch zur Laufzeit neue Nodes erzeugt werden. Die Struktur dient als eine Art Katalog, in dem alle Nodes enthalten sind, die der Nutzer zur visuellen Programmierung verwenden kann.
In Abbildung 5.3 wird in Auszügen die Klassendefinition des BaseNode und

[2]Vgl. http://www.github.com/benben/datasynth

```
struct Factory {
  typedef std::map<
            std::string,
            boost::function<NodePtr(int, float, float, string)>
          > FactoryMap;

  FactoryMap f;

  Factory() {
    using boost::phoenix::new_;
    using boost::phoenix::construct;
    using namespace boost::phoenix::arg_names;
    f["CSVParser"]    =
    construct<NodePtr>(new_<ds::CSVParser>(arg1, arg2, arg3, arg4));
    f["Render"]       =
    construct<NodePtr>(new_<ds::Render>(arg1, arg2, arg3, arg4));
    f["Point"]        =
    construct<NodePtr>(new_<ds::Point>(arg1, arg2, arg3, arg4));
    }

  NodePtr operator()
  (int ID, std::string const & type, float x, float y, string name) const {
    FactoryMap::const_iterator it = f.find(type);
    if (it == f.end()) throw bad_type_exception();
      return it->second(ID, x, y, name);
  }
};
```

Abbildung 5.2: Erzeugung eines Node-Objektes aus einer Liste aller vorhandenen Nodes

```
class BaseNode
{
  public:
    BaseNode();
    virtual ~BaseNode();

    string name;
    string type;
    int ID;

    vector<Pin*> input;
    vector<Pin*> output;
    virtual void init();
    virtual void process();
    virtual void basedraw();
    virtual void draw();
};

class Add : public BaseNode
{
  public:
    Add(int _ID, float _x, float _y, string _name);
    ~Add();
    void process();
};
```

Abbildung 5.3: Klassendefinition der BaseNode und der Node für die Addition als Beispiel

```
void Core::update() {
  BOOST_FOREACH(NodePtr node, nodes)
    node->process();

  for(unsigned int i = 0; i < nodes.size(); i++) {
      if(nodes[i]->bIsInvalid) {
        nodes.erase(nodes.begin()+i);
      }
    }
}

void Core::draw() {
  BOOST_FOREACH(NodePtr node, nodes) {
    if(!node->bIsInvalid) {
      node->draw();
    }
  }
}

void Core::handleMenuEvent(menuEventType & args) {
  if(args.handler == "CreateNode") {
    NodePtr temp =
    factory(ID++, args.valueType, Menu::Get()->x, Menu::Get()->y, args.value);
    temp->type = args.valueType;
    nodes.push_back(temp);
  }
}
```

Abbildung 5.4: Erzeugung von Objekten einer Klasse bestimmt durch eine Zeichenkette.

des Node für die Addition zweier Gleitkommazahlen als Beispiel dargestellt. Ein Node ist eine einfache C++ Klasse, die alle Eigenschaften der Klasse BaseNode erbt. Der Node für die Addition zweier Gleitkommazahlen überschreibt die process Methode der BaseNode, um zwei Input-Pins zu addieren. Andere Klassen, die von dem BaseNode erben, können die process Methode mit ihrer eigenen Funktionalität überschreiben. Das macht es möglich, alle Funktionen, die in einer C++ Methode implementiert werden können, in einer Klasse als Node zu kapseln und für die visuelle Programmierung von Datasynth zu verwenden.

In Abbildung 5.4 werden einzelne Ausschnitte aus dem Hauptprogramm gezeigt. In diesem Teil von Datasynth läuft die Hauptschleife ab, die alle Benutzereingaben behandelt, die Aktualisierung und Berechnung aller Nodes durchführt und Nodes und Connections hinzufügt oder löscht.

5.3 Implementierte Nodes

In Tabelle 5.1 werden implementierte Nodes exemplarisch aufgeführt und ihre Funktion beschrieben. Diese Nodes reichen bereits aus, um einfache Visualisierungen durchzuführen. Eine vollständige Liste aller für Datasynth implementierten Nodes findet sich im Anhang in Abschnitt 8.2.

Die Tabelle 5.2 zeigt, wie die grafischen Variablen aus Abschnitt 2.5 in Datasynth umgesetzt wurden. In dieser Tabelle werden ausschließlich Nodes beschrieben, die eine Ausgabe erzeugen. Die visuellen Variablen, Textur und Rotation wurden aufgrund der begrenzten Bearbeitungszeit nicht implementiert.

Node	Funktion
Add	Addiert zwei Spreads miteinander und gibt ein Spread als Summe dieser zurück.
Substract	Zieht vom Spread, der am ersten Input-Pin anliegt, den Spread vom zweiten Input-Pin ab und gibt die Differenz als Spread aus.
Multiply	Multipliziert zwei Spreads und gibt das Produkt als Spread aus.
Divide	Teilt den Spread am ersten Input-Pin durch den Spread der am zweiten Input-Pin anliegt und gibt das Ergebnis als Spread aus.
Variable	Diesem Node kann der Benutzer einen skalaren Wert zuweisen. Damit wird am Output-Pin ein Spread mit einer einzigen Variable erzeugt. Diese kann genutzt werden, um Berechnungen mit anderen Spreads durchzuführen.
CSVParser	Mit diesem Node hat der Benutzer die Möglichkeit, eine CSV-Datei zu importieren. Die „Spalten“ werden dabei als Spreads über die Output-Pins ausgegeben.
RGBColor	Erzeugt aus drei Input-Pins für die Farbkanäle Rot, Grün und Blau einen Spread mit Farbwerten. Dieser kann von anderen Nodes benutzt werden, um ausgegebene grafische Elemente einzufärben.
Line	Dieser Node zeichnet eine Linie. Der Node hat fünf Input-Pins, die Anfangskoordinaten, Endkoordinaten und Farbe der Linie bestimmen.
Circle	Dieser Node zeichnet einen Kreis. Der Kreis ist bestimmt durch Koordinaten, Radius und Farbe. Der Node hat dafür drei Input-Pins.
Render	Dieser Node erzeugt ein weiteres Fenster, in dem der Benutzer die aktuelle Ausgabe überprüfen kann. Wenn er mit der aktuellen Ausgabe zufrieden ist, kann er durch Tastendruck den Render-Node veranlassen, ein hochauflösende Bilddatei zu erstellen.

Tabelle 5.1: Exemplarische Auflistung implementierter Nodes

grafische Variable	**Umsetzung mit Datasynth**
Position in der Ebene	Jeder Node hat Input-Pins für die x- und y-Koordinaten der darzustellenden grafischen Primitive, wie beispielsweise Kreise und Rechtecke.
Größe	Nodes haben die Möglichkeit, die Größe der auszugebenden Elemente anzupassen. So kann beispielsweise bei der Verwendung des Circle-Node der Radius frei gewählt werden.
Helligkeitswert	Nodes haben die Möglichkeit eine Farbe anzunehmen. Durch die Farbe kann außerdem der Helligkeitswert festgelegt werden.
Textur	Textur ist nicht implementiert.
Farbe	Es gibt Nodes, die aus drei Eingangswerten eine Farbe im RGB- oder HSV-Farbraum erzeugen. Mit diesen Nodes kann die Farbgebung anderer Nodes beeinflusst werden.
Richtung	Objektrotation ist nicht implementiert.
Form	Es existieren unterschiedliche Nodes für unterschiedliche Formen, zum Beispiel Kreis, Rechteck, Punkt und Linie. Diese Nodes können kombiniert werden, um komplexe Formen zu realisieren.

Tabelle 5.2: Visuelle Variablen und ihre Umsetzung in Datasynth

	A	B	C	D	E	F	G	H
1	Stadtteil	lat	lon	R	G	B	Anzahl	führende Branche
	Südvorstadt	51.3224855	12.3746133	165	239	0	63	Filmwirtschaft
2	Zentrum-Süd	51.3332651	12.3738194	201	0	122	61	Werbemarkt
3	Zentrum	51.339997	12.3762651	0	163	131	57	Buchmarkt
4	Zentrum-Südost	51.3331504	12.3852492	0	163	131	55	Buchmarkt
5	Zentrum-West	51.3362214	12.3571139	201	0	122	47	Werbemarkt
6	Zentrum-Ost	51.3419162	12.3887594	201	0	122	41	Werbemarkt
7	Plagwitz	51.3281994	12.3382998	201	0	122	40	Werbemarkt
8	Zentrum-Nordwest	51.3499642	12.3570389	0	163	131	31	Buchmarkt
9	Gohlis-Süd	51.3622477	12.3622622	0	163	131	30	Buchmarkt
10								
11	Zentrum-Nord	51.35077	12.37796	255	100	0	29	Software/Games-Industrie

Abbildung 5.5: Ausgangsdaten in Tabellenform (Ausschnitt)

5.4 Beispielanwendung

In diesem Kapitel wird mit Datasynth eine Informationsvisualisierung durchgeführt. Die Ausgangsdaten aus Abbildung 5.5 wurden aus dem API.Leipzig Projekt gewonnen und in Tabellenform aufbereitet. Diese Tabelle wurde als CSV-Datei exportiert. In Abbildung 5.6 werden Ausschnitte aus der Benutzeroberfläche von Datasynth gezeigt. Diese zeigen exemplarisch Nodes, die für die Informationsvisualisierung benutzt werden. Zuerst wird die CSV-Datei mithilfe eines CSVParser-Node importiert. Danach werden unterschiedliche Berechnungen durchgeführt um die Ausgabe des Circle-Nodes zu beeinflussen. Ein Render-Node erzeugt die Ausgabe und ermöglicht eine Exportfunktion. Eine vollständige Übersicht der Benutzeroberfläche für diese Anwendung findet sich im Anhang in Abschnitt 8.4. In Abbildung 5.7 wird ein Ausschnitt aus der fertigen grafischen Darstellung gezeigt. Die komplette Darstellung findet sich im Anhang in Abschnitt 8.5.

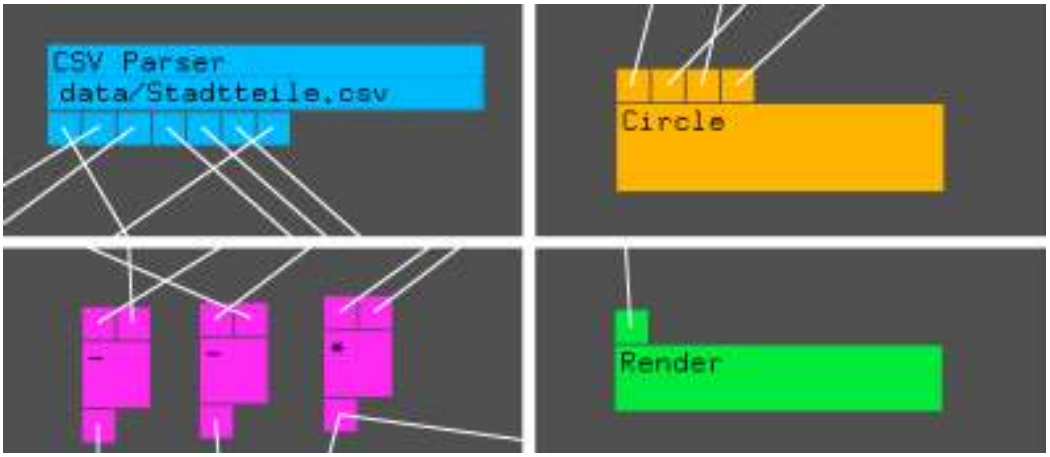


Abbildung 5.6: Ausschnitte aus der Benutzeroberfläche. Oben links: CSVParser-Node importiert die Daten aus einer CSV-Datei. Unten links: verschiedene Nodes zur Berechnung. Oben rechts: Circle-Node erzeugt Kreise an den gegebenen Positionen. Unten rechts: Render-Node erzeugt eine Ausgabe.

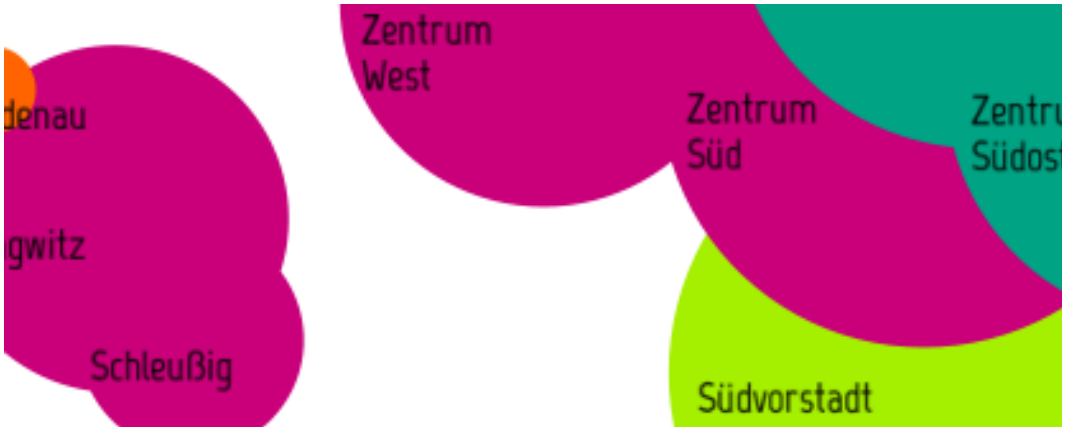


Abbildung 5.7: Ausschnitt aus der grafischen Darstellung

5.5 Probleme

Während der Realisierung von Datasynth traten einige Probleme auf, die im Folgenden erläutert werden.

Da C++ eine sehr maschinennahe Sprache ist und auf Performanz optimiert ist, ist es schwierig Reflections und Introspections durchzuführen. Es können während der Laufzeit keine Informationen über die Beschaffenheit einer Klasse abgefragt werden. Das können zum Beispiel die Ausgabe aller vorhandenen Methoden und Instanzvariablen eines Objektes sein. Diese Eigenschaften können außerdem nicht dynamisch verändert werden. Beispielsweise ist es nicht möglich, die process Methode einer Node zur Laufzeit zu überschreiben. Um trotzdem flexibel Nodes zur Laufzeit erzeugen zu können, erben alle Nodes von einer universellen Elternklasse. In einem Container werden dann ausschließlich Zeiger auf die Objekte vom Typ der Elternklasse gespeichert. An den Speicherstellen liegen dann die eigentlichen Objekte einer Node-Klasse. Das Hauptprogramm kann über alle Elemente des Containers iterieren und somit alle vom Benutzer erzeugten Nodes aufrufen.

Universelle Datentypcontainer sind in C++ nicht standardmäßig vorhanden.[3] Um die Verbindungen zwischen den einzelnen Nodes datentypagnostisch zu halten, wurde ein eigener Datentyp Spread definiert.[4] Durch die Abstraktion in einen boost::variant Container ergibt sich eine einheitliche Möglichkeit, primitive Datentypen, wie beispielsweise Gleitkommazahlen und Zeichenketten abzuspeichern. Vor der Verwendung in einer Node werden diese Datentypen

[3] Nullzeiger sind möglich, erfordern aber eine unsichere und unflexbile Wiederherstellung des ursprünglichen Datentyps.

[4] Eine Definition des Begriffs „Spread“ erfolgte im Abschnitt 4.3

mit boost::get rekonstruiert. Der Vorteil gegenüber Nullzeigern ist, dass bei fehlgeschlagener Wiederherstellung eine Fehlerbehandlung möglich ist.

openFrameworks unterstützt nicht das Erstellen und Verwenden mehrerer unabhängiger Fenster. Deshalb musste auf ein Add-on zurückgegriffen werden, welches die Fensterverwaltung mit einem alternativen System austauscht.[5] Damit können nach dem Start des Programms neue Fenster erzeugt werden, was beispielsweise für die Funktionalität des Render-Node wichtig ist.

[5] openFrameworks verwendet standardmäßig „GLUT“ zur Fensterverwaltung. Das für Datasynth verwendete „ofxFenster“ Add-on benutzt dagegen die neuere und flexiblere Bibliothek „GHOST“ aus dem Blender-Projekt.

6 Evaluation

In diesem Kapitel wird Datasynth anhand der in Abschnitt 4.2 aufgestellten Anforderungen bewertet.

Durch die Bedienung von Datasynth mit visueller Programmierung muss keine textuelle Programmiersprache erlernt werden. Stattdessen kann der Nutzer intuitiv grafische Elemente auf der Oberfläche erzeugen und durch Verbindungen eine Programmlogik erstellen. Das Ergebnis einer Berechnung oder die Ausgabe von Grafiken erfolgt dabei unmittelbar in einem seperaten Fenster. Dort kann der Nutzer ohne einen Zwischenschritt, wie beispielsweise Kompilierung oder Ausführung, die grafische Darstellung sofort überprüfen und gegebenenfalls anpassen. Dadurch wird eine *einfache Benutzerinteraktion* ermöglicht.

In Abschnitt 5.3 wurden Nodes aufgelistet, die für die prototypische Realisierung entstanden sind. Damit können erste Informationsvisualisierungen[1] durchgeführt werden. Aufgrund der begrenzten Bearbeitungszeit konnten zwei visuelle Variablen nicht implementiert werden. Der sehr komplexe Node für die Texturierung und die Funktion der Rotation wurden nicht umgesetzt. Die feh-

[1] Vgl. Abschnitt 8.3 und 8.5 im Anhang

lende Funktionalität kann aber durch den modularen Aufbau von Datasynth nachträglich hinzugefügt werden.

Für die *flexible Ein- und Ausgabe* wurde jeweils eine Möglichkeit implementiert, um die komplette Prozesskette von Datasynth umzusetzen. Es existiert ein Node, der eine CSV-Datei einliest und entsprechend der Anzahl der vorhanden Spalten automatisch Output-Pins erzeugt, an denen die einzelnen Spalten als Spreads ausgegeben werden. Für die Ausgabe als Bild existiert ein Render-Node. Dieser erzeugt ein Ausgabefenster in variabler Größe und kann per Tastendruck hochauflösende Bilddateien im PNG-Format exportieren.

Die *Performanz* von Datasynth war bei der Erstellung aller für diese Arbeit durchgeführten Informationsvisualisierungen sehr gut. Das Grundsystem wurde so programmiert, dass Nodes die process Methode nur ausführen, wenn die Werte an den Input-Pins verändert werden. Somit werden komplexe Rechenaufgaben nur durchgeführt, wenn der Nutzer Parameter anpasst. Die grafische Ausgabe des Render-Node erfordert den größten Rechenaufwand, da die Ausgabe kontinuierlich gezeichnet wird. Dieser Aufwand kann durch entsprechende Steuerung reduziert werden.

Alle Nodes sind einfache C++ Klassen, die von einer Elternklasse alle Variablen und Methoden erben, die für die Verwendung in Datasynth nötig sind. Somit ist eine hohe *Erweiterbarkeit* gegeben, da es möglich ist, nahezu jede Funktionalität, die in einer C++ Klasse implementiert werden kann, als Node in Datasynth nutzbar zu machen. Auf eine eingebettete Skriptsprache wurde aus zeitlichen Gründen verzichtet.

Durch die Verwendung von openFrameworks und boost als Basis der Entwicklung ist es möglich, Datasynth auf allen Plattformen auszuführen, die von

openFrameworks unterstützt werden. Das sind Linux, Mac OS X, Microsoft Windows. Dadurch ist Datasynth *plattformunabhängig* einsetzbar.

Datasynth wird öffentlich auf der Plattform GitHub entwickelt.[2] Als Lizenz wurde die MIT-Lizenz gewählt. Diese ermöglicht jedem Benutzer eine offene, aber auch kommerzielle Weiterentwicklung.

Datasynth kann neben den in Abschnitt 5.5 genannten Problemen nahezu alle Anforderungen erfüllen. Trotzdem ist Datasynth nicht für den produktiven Einsatz geeignet, da wichtige Hilfestellungen für den Benutzer, wie beispielsweise ein Handbuch, fehlen. Im Anhang in Abschnitt 8.5 befindet sich eine Informationsvisualisierung, die mit Datasynth realisiert wurde.

[2] Vgl. Abschnitt 5.1

7 Zusammenfassung

7.1 Fazit

Die Konzeption und prototypische Realisierung einer Software für die Informationsvisualisierung hat gezeigt, dass es sinnvoll ist, ein auf dieses Gebiet zugeschnittenes Softwarekonzept zu entwickeln. Die Anforderungen sind dabei stark anwendungsabhängig. Es müssen viele unterschiedliche Datenformen mit den visuellen Variablen der grafischen Darstellung verbunden werden können. Die Automatisierung dieses Transkriptionsprozesses ist sinnvoll, wenn der Benutzer diesen Ablauf umfassend parametrisieren, steuern und gestalten kann. Datasynth bietet dafür bereits unterschiedliche Möglichkeiten und macht eine grafische Darstellung, die dem Umfang des Initialbeispiels entspricht, möglich. Für die Erzeugung von zusätzlichen Informationen wie beispielsweise Beschriftungen, Skalen oder Maßstäbe existieren nur rudimentäre Ansätze. Fehlende Funktionalität kann aber vom Benutzer hinzugefügt werden, ohne den Kern der Software ändern zu müssen. Datasynth besitzt hierfür ein universelles Node-System, welches modulartig erweiterbar ist. Es besteht damit die Möglichkeit auf zukünftige veränderte Anforderungen an Informationsvisualisierungen flexibel zu reagieren.

Die Steuerung von Datasynth mithilfe visueller Programmierung gibt dem Benutzer die intuitive Möglichkeit, ästhetisch ansprechende Darstellungen anzufertigen, ohne selbst Quellcode zu schreiben. Um die visuelle Programmierung von Datasynth noch effizienter zu machen, müssen aber noch einige Verbesserungen an der Benutzeroberfläche stattfinden.

Der generische Ansatz der Software vermeidet es, Vorgaben bei der Gestaltung von grafischen Darstellungen zu machen. Dadurch behält der Benutzer ein breites Spektrum an Gestaltungsfreiheiten. Das erfordert aber auch die Kenntnis und Anwendung der Gestaltungsregeln. Der Benutzer muss selbst entscheiden, welche visuellen Variablen sich für eine bestimmte Aufgabe eignen. Ein System, welches dem Benutzer Hilfestellungen in diesem Bereich gibt, muss noch erforscht werden.

Datasynth stellt nur einen Prototyp dar, kann aber für den produktiven Einsatz weiterentwickelt werden, um einem vereinfachten Umgang mit der Informationsvisualisierung zu ermöglichen. Davon profitieren vor allem Gelegenheitsanwender, die die Informationsvisualisierung als Werkzeug für ihre jeweiligen Arbeitsfelder einsetzen. Zum Beispiel können Journalisten die Software nutzen um Texte mit grafischen Darstellungen anzureichern. Aber auch Wissenschaftler erhalten damit eine einfache und schnelle Möglichkeit erste Erkundungen einer Datenmenge durchzuführen. Nicht zuletzt kann es Künstlern die Möglichkeit geben, auf Basis von Daten ästhetisch ansprechende Informationsvisualisierungen zu erschaffen. Damit kann die Software in unterschiedlichen Bereichen unseres Lebens genutzt werden und zu einem besseren Verständnis der digitalen Welt beitragen.

7.2 Ausblick

In diesem Abschnitt werden vielversprechende Ansätze beschrieben, die für die Weiterentwicklung der Konzeption und des Prototyps genutzt werden können. Einige Möglichkeiten bedingen weiterer Forschungsarbeit und können als Thesen betrachtet werden.

Skriptsprache Es können universelle Nodes geschaffen werden, die vom Benutzer selbst mit einer höheren Skriptsprache programmiert werden können. Fehlende Funktionalitäten können damit vom Benutzer selbst implementiert werden. Die Anwendung einer Skriptsprache in der eigentlichen Software ist von Vorteil, um die in Abschnitt 5.5 genannten Probleme zu umgehen. Dabei sollte nur die rechenintensive Programmlogik in C++ weiterentwickelt werden.

Gruppierung von Nodes Um schnell und effizient komplexe, grafische Darstellungen generieren zu können, ist eine Gruppierung von Nodes sinnvoll. Damit können Templates erzeugt werden, die eine grafische Darstellungsform in einem Node kapseln. So sind einzelne Nodes für Tortendiagramme oder Netzwerkstrukturen denkbar.

Weitere Ausgabemöglichkeiten Zusätzlich zu der Ausgabe als statische Bilddatei sind weitere Ausgabeformate sinnvoll. So können Nodes integriert werden, die eine Ausgabe als Postscript oder in 3D-Austauschformate ermöglichen. Um die grafische Weiterverarbeitung der erzeugten Darstellungen effizienter zu machen, bietet sich eine Exportfunktion für einzelne Teilbereiche an. Beispielsweise können aus einem erzeugten Bild einzelne Ebenen exportiert werden.

Interaktionsmöglichkeiten Zusätzlich zu einer rein statischen Darstellung können mit Datasynth Interaktionsmöglichkeiten erzeugt werden. Eine automatische Ausgabe als interaktive HTML5-Website ist denkbar, bei der einzelne Elemente anklickbar und durch Benutzerinteraktion veränderbar sind. Außerdem sind Zoomfunktionen sinnvoll, um dem Betrachter weitere Möglichkeiten zur Erkundung der grafischen Darstellung zu geben.

Evaluation Datasynth kann dem Nutzer behilflich sein, die richtige Form der grafischen Darstellung zu finden und ihn mit adäquaten Mitteln darauf hinweisen, welche Darstellungen sich für einen gegebenen Datensatz besonders gut eignen.

8 Anhang

8.1 Liste von bestehenden Softwarelösungen

Name	Webadresse
d3	http://mbostock.github.com/d3/
Gephi	http://gephi.org/
GGobi	http://www.ggobi.org/
Graphviz	http://www.graphviz.org/
InfoVis Toolkit	http://ivtk.sourceforge.net/
MAX 6	http://cycling74.com/
MeVisLab	http://mevislab.de/
N-Land	http://davis.wpi.edu/ matt/courses/nland/cgi93.html
NodeBox 2 Beta	http://beta.nodebox.net/
prefuse	http://prefuse.org/
Processing	http://processing.org/
protovis	http://mbostock.github.com/protovis/
Pure Data	http://www.puredata.info/
VTK	http://www.vtk.org/
vvvv	http://www.vvvv.org/

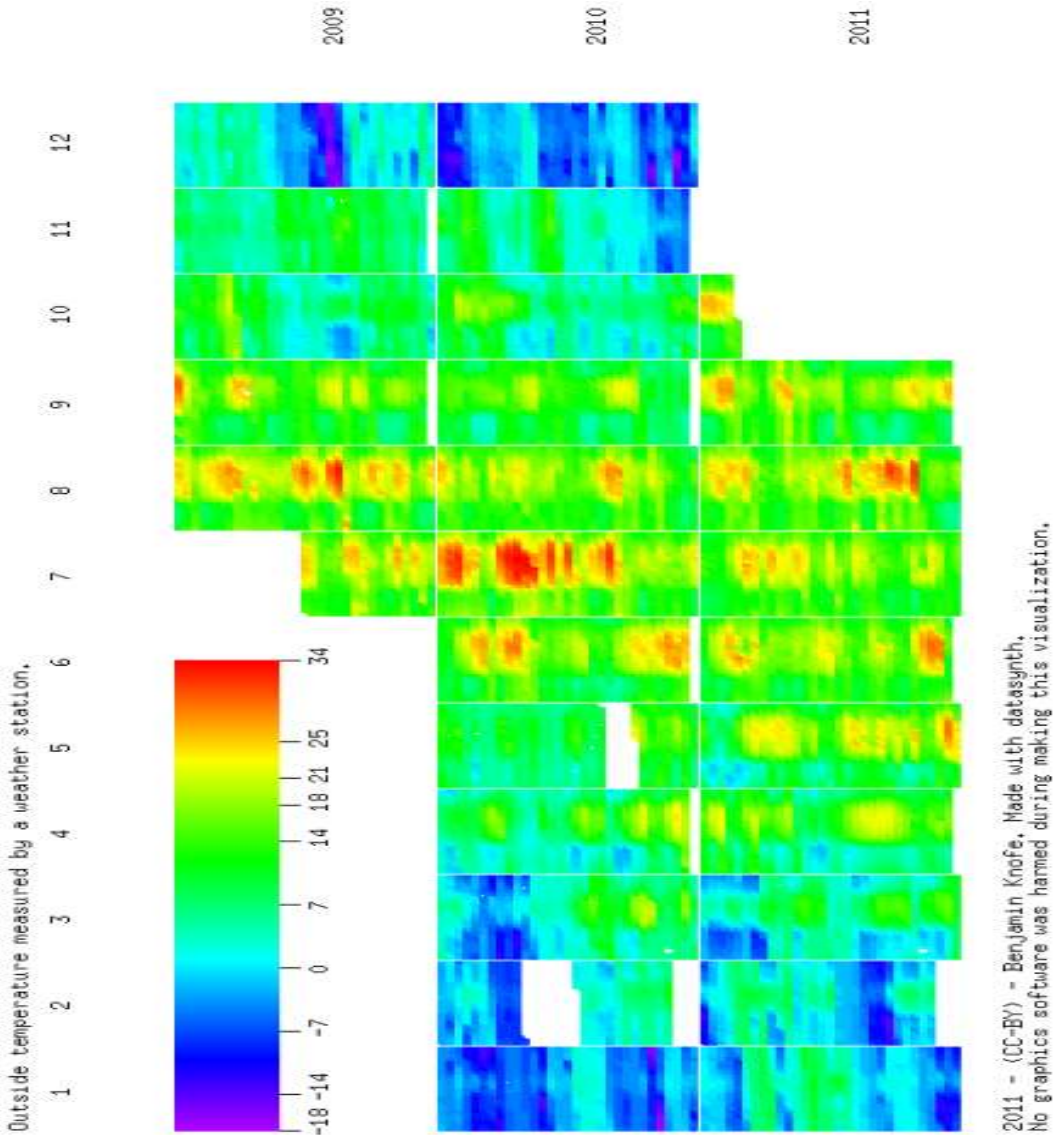
8.2 Liste aller implementierten Nodes

Node	**Funktion**
Add	Addiert zwei Spreads miteinander und gibt ein Spread als Summe dieser zurück.
Substract	Zieht vom Spread, der am ersten Input-Pin anliegt, den Spread vom zweiten Input-Pin ab und gibt die Differenz als Spread aus.
Multiply	Multipliziert zwei Spreads und gibt das Produkt als Spread aus.
Divide	Teilt den Spread am ersten Input-Pin durch den Spread, der am zweiten Input-Pin anliegt und gibt das Ergebnis als Spread aus.
Modulo	Berechnet den Rest der Division zweier Spreads und gibt diesen als Spread aus.
GreaterThan	Vergleicht zwei Spreads miteinander. Wenn der Spread am ersten Input-Pin größer als der am zweiten Input-Pin ist, wird 1 zurückgegeben.
LessThan	Vergleicht zwei Spreads miteinander. Wenn der Spread am ersten Input-Pin kleiner als der am zweiten Input-Pin ist, wird 1 zurückgegeben.
Min	Gibt den kleinsten Wert eines Spreads zurück.
Max	Gibt den größten Wert eines Spreads zurück.
Map	Verschiebt die Werte eines Spreads von einem Wertebereich in einen anderen.
Unique	Entfernt alle Duplikate eines Wertes aus einem Spread.

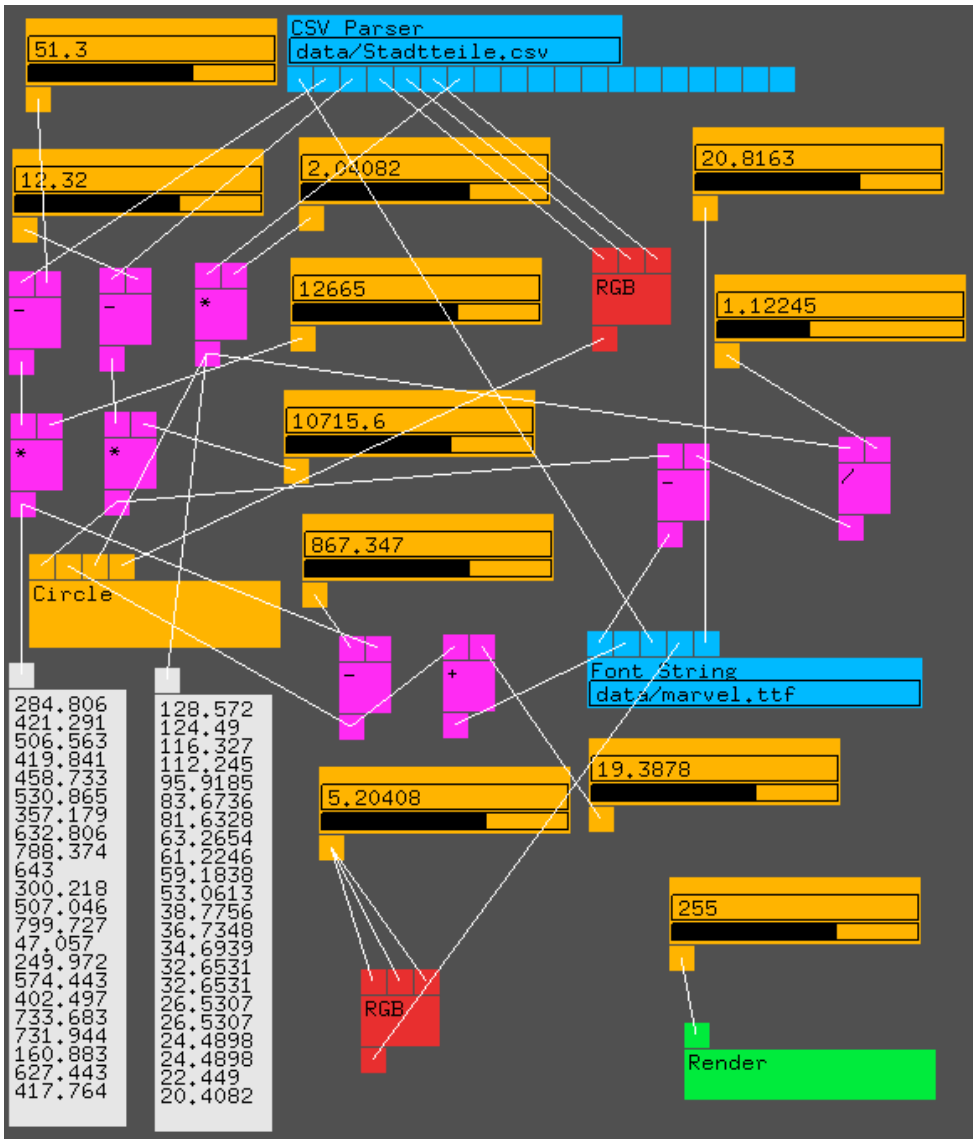
Node	Funktion
OutBox	Zeigt den Inhalt eines Spreads für den Benutzer an. Dieser Node hat keine Funktion für den Programmablauf, sondern dient dem Nutzer zur Kontrolle von Spreads.
Variable	Diesem Node kann der Benutzer einen skalaren Wert zuweisen. Damit wird am Output-Pin ein Spread mit einer einzigen Variable erzeugt. Diese kann genutzt werden, um Berechnungen mit anderen Spreads durchzuführen.
CSVParser	Mit diesem Node hat der Benutzer die Möglichkeit, eine CSV-Datei zu importieren. Die „Spalten“ der Daten werden dabei als Spreads ausgegeben.
RGBColor	Erzeugt aus drei Input-Pins für die Farbkanäle Rot, Grün und Blau einen Spread mit Farbwerten. Dieser kann von anderen Nodes benutzt werden, um ausgegebene grafische Elemente einzufärben.
Text	Gibt dem Nutzer die Möglichkeit, Text einzugeben und als Spread weiterzuverarbeiten.
String	Zeichnet einen einfachen Text in der Ausgabe. Die Ausgabe des Textes kann durch die Position und die Farbe beeinflusst werden.
Line	Dieser Node zeichnet eine Linie. Der Node hat fünf Input-Pins, die Anfangskoordinaten, Endkoordinaten und Farbe der Linie bestimmen.
Circle	Dieser Node zeichnet einen Kreis. Der Kreis ist bestimmt durch Koordinaten, Radius und Farbe. Der Node hat dafür drei Input-Pins.
Rectangle	Dieser Node zeichnet ein Rechteck. Das Rechteck ist bestimmt durch Koordinaten, Breite, Höhe und Farbe.
Render	Dieser Node erzeugt ein weiteres Fenster, in dem der Benutzer die aktuelle Ausgabe überprüfen kann. Wenn er mit der aktuellen Ausgabe zufrieden ist, kann er durch Tastendruck den Render-Node veranlassen, ein hochauflösende Bilddatei zu erstellen.

Tabelle 8.1: Alle in Datasynth implementieren Nodes

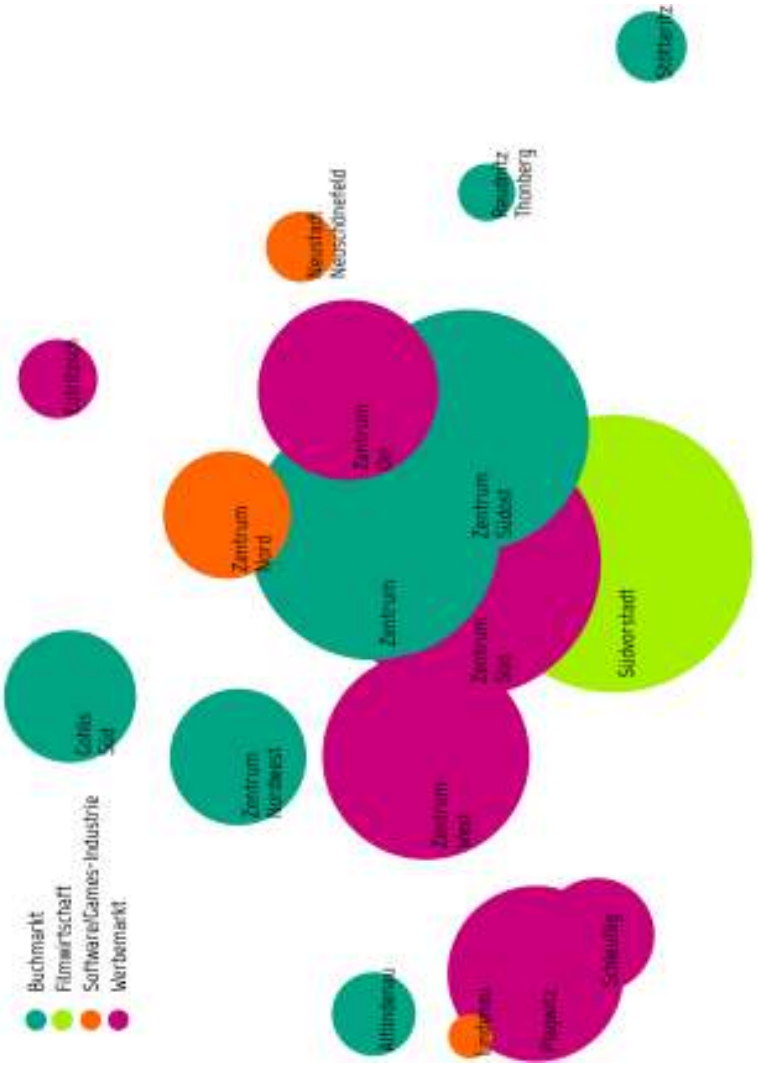
8.3 Beispiel: Temperaturen einer Wetterstation



8.4 Beispiel: Benutzeroberfläche



8.5 Beispiel: Medienunternehmen in den Stadtteilen Leipzigs



9 Literaturverzeichnis

[Bertin 1974] Bertin, Jacques: *Graphische Semiologie - Diagramme, Netze und Karten*. Berlin : Walter de Gruyter, 1974

[Card u. a. 1999] Card, Stuart K. ; Mackinlay, Jock D. ; Shneiderman, Ben: *Readings in Information Visualization*. San Francisco : Morgan Kaufmann Publishers, 1999

[EMC 2011] *IDC Digital Universe Studie*. 2011. – URL `http://germany.emc.com/about/news/press/2011/20110628-01.htm`. – Letzter Abruf 18.08.2011

[Fry 2004] Fry, Benjamin J.: *Computational Design*, Massachusetts Institute of Technology, Dissertation, 2004

[Gabler 2011] *Gabler Wirtschaftslexikon*. 2011. – URL `http://wirtschaftslexikon.gabler.de/Archiv/54483/daten-v5.html`. – Letzter Abruf 18.08.2011

[Gitta 2011] *thematische Karten*. 2011. – URL `http://www.gitta.info/ThematicCart/de/html/Einfuehrung_learningObject1.html`. – Letzer Abruf 18.08.2011

[Henning und Vogelsang 2007] Henning, Peter A. ; Vogelsang, Holger: *Handbuch Programmiersprachen*. München : Hanser, 2007

[Höher 2011] Höher, Peter A.: *Grundlagen der digitalen Informationsübertragung*. Wiesbaden : Vieweg + Teubner, 2011

[Informationsbegriff 2011] *Information*. 2011. – URL `http://de.wikipedia.org/wiki/Information`. – Letzer Abruf 18.08.2011

[Keim 2002] Keim, Daniel A.: Information Visualization and Visual Data Mining. In: *IEEE Transactions on Visualization and Computer Graphics* 7 (2002), Nr. 1

[Mashup 2011] *Mashup*. 2011. – URL `http://de.wikipedia.org/wiki/Mashup_(Internet)`. – Letzer Abruf 24.08.2011

[McCandless 2011] *Creation of the Book*. 2011. – URL `http://de.wikipedia.org/wiki/Mashup_(Internet)`. – Letzer Abruf 13.09.2011

[McCandless 2009] McCandless, David: *Information is Beautiful*. London : Collins, 2009

[openFrameworks FAQ 2011] *faq of openFrameworks*. 2011. – URL `http://www.openframeworks.cc/about/faq`. – Letzer Abruf 19.09.2011

[openFrameworks Geschichte 2011] *history of openFrameworks*. 2011. – URL `http://www.openframeworks.cc/about`. – Letzer Abruf 19.09.2011

[openFrameworks Wiki 2011] *openFrameworks Wiki*. 2011. – URL `http://wiki.openframeworks.cc/index.php?title=OF_code_structure_(image)`. – Letzer Abruf 19.09.2011

[Preim und Dachselt 2010] PREIM ; DACHSELT: *Interaktive Systeme - Band 1: Grundlagen, Graphical User Interfaces, Informationsvisualisierung*. Heidelberg : Springer, 2010

[Schumann und Müller 2000] SCHUMANN, Heidrun ; MÜLLER, Wolfgang: *Visualisierung - Grundlagen und allgemeine Methoden*. Berlin : Springer, 2000

[Schäfer 2010] SCHÄFER, Thomas: *Statistik I - Deskriptive und Explorative Datenanalyse*. Wiesbaden : VS Verlag, 2010

[Tufte 1990] TUFTE, Edward R.: *Envisioning Information*. Cheshire : Graphics Press, 1990

10 Tabellenverzeichnis

11 Abbildungsverzeichnis