Kapitel 1

Einleitung

METAPOST (MP) ist eine grafische Programmiersprache und ein Compiler/Interpreter mit PostScript-Ausgabe (Vektorgrafik). Die Sprache ist nicht imperativ sondern deklarativ und ermöglicht, die wesentlichen Komponenten von bestimmten benötigten grafischen Formen geeignet anzuordnen, zu zeichnen und mit LaTeX zu beschriften.

Historie von MP

1984 veröffentlichte Donald E. Knuth das Programmsystem METAFONT (MF) zum Entwurf von Schriften *(fonts)*, hervorragend dokumentiert im METAFONTbook [33]. MP wurde von John D. Hobby etwa ab 1990 auf der Basis von MF Version 1.9 bei AT&T Bell Laboratories entwickelt, nachdem er zuvor als Doktorand von Professor Knuth an der Stanford University bereits wesentlich zur Entwicklung von MF beigetragen hatte. Nach 1995 wurde MP frei zugänglich. Nachdem sich über lange Zeit zahlreiche Fehler angesammelt hatten, übernahm 2006 Taco Hoekwater die Betreuung und Weiterentwicklung von MP. Neben der Fehlerbeseitigung, erweiterte er MP um eine Library, eine dynamische Speicherverwaltung und um den Übergang von Pascal WEB auf CWEB, endend in Version 1.5. Ab 2009 folgte als weitere Neuerung eine wesentliche Verbesserung der numerischen Genauigkeit. Außer der bisherigen 32 Bit-Festpunktarithmetik steht nun wahlweise auch eine 64 Bit-Gleitpunktarithmetik (IEEE Floating Point Arithmetic) und bei Bedarf eine Arithmetik auf binärer oder dezimaler Basis mit fast beliebig wählbarer Stellenzahl zur Verfügung. Dies führte zu der seit 1.6.2019 verfügbaren Version 2.00, sodass mit MP ein modernes, sehr stabiles Grafiksystem von hoher Qualität zur Verfügung steht. Die weitere Betreuung von MP hat jetzt Luigi Scarso übernommen.

Die Homepage von MP ist `tug.org/metapost`. Dort findet man auch die Dokumentation [25] und viele interessante Veröffentlichungen, Tutorials [19, 20] und umfangreiche

Beispielsammlungen [44, 57], deren Lektüre ich sehr empfehlen kann. Viele Anregungen enthält auch das MetaFun-Manual von Hans Hagen [14]. Die Dokumentation [25] steht auch lokal mit `texdoc metapost` zur Verfügung.

Viele nützliche Zusatzpakete findet man auf dem CTAN-Server:
`CTAN graphics/metapost/contrib/macros/`

Die derzeitige Entwickler-Homepage von MP ist
`foundry.supelec.fr/projects/metapost/`.

Mailing-List: `tug.org/mailman/listinfo/metapost`

Eigenschaften von MP

MP und MF unterscheiden sich vor allem hinsichtlich des Ausgabeformats: MF gibt das Ergebnis im Pixel-Format (GF, *generic font*) aus, MP dagegen als Vektorgrapfik im PostScript-Format. Außerdem wurde MP um eine komfortable Beschriftung (*labeling*) mit TEX/LATEX-Text und die Verwendung von Farben erweitert. Die Zeicheneinheit ist nicht ein Pixel sondern ein *big point* bp (PostScript-Punkt). Ferner gibt es in MP gestrichelte Linien und Pfeile sowie höhere bildverarbeitende Operationen wie `clip`, `image`, `buildcycle`, `thelabel`.

MP kennt wie jede Programmiersprache arithmetisch-logische Ausdrücke und elementare mathematische Funktionen sowie Kontrollstrukturen zur Steuerung des Programmflusses. Die besondere Leistungsfähigkeit als grafische Programmiersprache liegt jedoch in der Bereitstellung von speziellen Variablentypen zur adäquaten Definition und Beschreibung der grafischen Grundelemente Punkt, Linie, Fläche und Farbe und entsprechender Operationen zur Transformation, Platzierung und zum Zeichnen dieser Objekte. Sehr wichtig ist dabei, dass alle schrittweise gezeichneten Bildteile gespeichert bleiben bis zur endgültigen Fertigstellung einer Zeichnung. Dadurch kann man stets auf alle Bildteile zugreifen, z. B. um Schnittpunkte zu berechnen oder Beschriftungen exakt zuzuordnen. Eine weitere Stärke von MP ist die sehr leistungsfähige Definition von Unterprogrammen (Makros) zur flexiblen Erweiterung der Funktionalität. Von besonders hoher Qualität ist auch das von Donald E. Knuth und John D. Hobby speziell entwickelte Interpolationsverfahren zum Zeichnen gekrümmter Kurvenverläufe.

Einbetten von MP-Code in ein LATEX-Dokument

Im Normalfall erzeugt der MP-Interpreter aus einer MP-Quelldatei eine Bilddatei im PostScrip-Format, die dann in ein LATEX-Dokument mit dem Befehl `\includegraphics` des Pakets graphicx eingefügt werden kann. Es besteht jedoch häufig der Wunsch, den MP-Quellcode direkt in das LATEX-Dokument einzubetten und das gesamte Dokument (Text und Bilder) möglichst in nur einem Bearbeitungsschritt zu formatieren.

Im Folgenden geben wir eine Übersicht über einige Pakete, die zur Lösung dieses Einbettungsproblems entwickelt wurden, ohne Anspruch auf Vollständigkeit. Alle aufgeführten Pakete stehen auf CTAN zur Verfügung und sind in TEX Live enthalten. Die jeweilige Dokumemtation erhält man mit dem Kommando `texdoc <paketname>`.

emp Das Paket *EMP—Encapsulated Metapost for LATEX* von Thorsten Ohl, [43], erschien 1997. Es ist wohl eines der ersten Pakete und war richtungsweisend für die weitere

Entwicklung. Es erlaubt die Beschriftung mit TEX/LATEX-Text und die Verwendung des Graph-Pakets `graph.mp`.

mpgraphics Das Paket stammt von der Persischen TEX User Group [30] und wurde von Vafa Khalighi und Zal Mehran ab 2010 entwickelt. Es erfordert nur einen einzigen Bearbeitungsschritt.

gmp Das Paket *gmp—Enable integration between METAPOST pictures and LaTeX* von Enrico Gregorio [13] erschien 2011 und ist eine Weiterentwicklung von emp und mpgraphics, die als Neuerung eine enge Verbindung zwischen dem LATEX-Code des Dokuments und dem MP-Code der Bilder ermöglicht.

mpostinl Das Paket *mpostinl—Embed METAPOST figures within LATEX documents* von Niklas Beisert ist anfangs 2017 auf CTAN erschienen [3]. Durch die Verwendung von effizienten Optionen ist das Paket sehr flexibel anwendbar und kann an unterschiedliche Anforderungen angepasst werden. Dies erlaubt das Zusammenspiel mit einigen Paketen, das sonst nicht ohne Weiteres möglich wäre.

luamplib Mit der Entwicklung der neuen Engine LuaTEX und in Verbindung mit dem System ConTEXt von Hans Hagen, das MetaPost in Form von MetaFun als integralen Bestandteil enthält, hat das Einbettungsproblem neue Impulse und eine neue Struktur erhalten. Taco Hoekwater schuf dafür die Library LuaMPLib. Das Paket luamplib [15] ermöglicht deren Verwendung in LuaLATEX, wobei der Code im Wesentlichen aus ConTEXt stammt und entsprechend angepasst wurde. Das LuaTeX Development Team um Hans Hagen, Taco Hoekwater, Hartmut Henkel und Luigi Scarso entwickelt das Paket luamplib ständig weiter und fügt neue Funktionalität hinzu.

Das Thema Einbetten wiird in Kapitel 17 im Detail beschrieben.

Das Einbetten des MP-Codes der Bilder in ein LATEX-Dokument hat den Vorzug, dass man nur ein Dokument erstellen, bearbeiten und pflegen muss. Insbesondere bei einfachen Bildern ist dies vorteihaft, zumal ein Bild auch auf der Zeile (*inline*) in den Text eingefügt werden kann. Allerdings wird bei etwas komplizierten Bildern der zugehörige MP-Code unter Umständen sehr umfangreich, sodass man schnell den Überblick verlieren kann. Auch die Wiederverwendbarkeit der Bilder in anderen Dokumenten ist nicht offensichtlich und das Einfügen des Listings des MP-Codes zusätzlich zum Bild erfordert weiteren Aufwand. Verlage von Fachzeitschriften verlangen oft die Bilder als separate, selbständige (*stand-alone*) Bilddateien im eps- oder pdf-Format mit korrekter Boundingbox. In diesen Fällen sollte man die Bilder nach den in Kapitel 5 vorgeschlagenen Methode getrennt entwerfen und mit dem Befehl `\includegraphics` des Pakets graphicx in das Dokument einzufügen.

Unter Umständen ist auch eine Kombination beider Vorgehensweisen denkbar, einfache Bilder eingebettet, insbesondere auch zur Verwendung auf der Zeile, und komplizierte getrennt zu entwerfen und standardmäßig einzubinden.

Dateiorganisation

Man muss jedoch sicherstellen, dass die Bilder und der entsprechende MP-Code (*listing*) im Dokument mit den entworfenen Grafiken stets konsistent bleiben. Für die Erstellung

dieses Buches habe ich die folgende Vorgehensweise und Dateiorganisation verwendet, die auch auf andere Projekte sinngemäß übetragbar ist.

Tabelle 1.1: Dateiorganisation.

		buch/		
		LaTeX-Hauptdokument `book.tex`		
		Stildateien		
		Kapiteldateien		
		Bibliografiedatei		
list/	**fig/**	**pdf/**	**code/**	**defs/**
Arbeitsverz. zum Entwurf der Bilder	eps-Dateien	Bilder pdf-Dateien	MP-Code	Makros
Shell-Skript	`<mpfile>.eps`	`<mpfile>.pdf`	`<mpfile>.mp`	`def_<Makroname>.mp`
Präambel				
Postambel				
`dummy.tex`				
Datendateien				
Programme				

Das LaTeX-Hauptdokument mit seinen Kapitel-, Stil- und Bibliografiedateien (*files*) befindet sich im Verzeichnis (*directory*) `buch` Es wird mit `lualatex`, `bibtex` und `biber` bearbeitet. Die resultierende pdf-Datei kann man mit einem pdf-Viewer anzeigen und gegebenenfalls ausdrucken.

Die Unterverzeichnisse `list`, `fig`, `pdf`, `code` und `defs` dienen zum Entwurf der Bilder in `list` und zur Archivierung in `fig`, `pdf` und `code`. Das Verzeichnis `defs` enthält die Makrosammlung. Der Entwurf der Bilder erfolgt ausschließlich im Arbeitsverzeichnis `list` mit Hilfe eines Shell-Skripts, das die Bearbeitungsschritte und Archivierung übernimmt. Der eigentliche Entwurfsprozess ist ein Wechsel zwischen Editor und diesem Shell-Skript. Alle Hilfsdateien wie `preamble.mp`, `postamble.mp`, `dummy.tex`, Datendateien der Beispiele etc. und die Makrosammlung, die das Shell-Skript auch benötigt, werden im Verzeichnis `list` und `defs` vorgehalten. Zur Sicherstellung der Konsistenz, werden die Bilder und ihr jeweiliger MP-Code in das Hauptdokument mit den Befehlen `\includegraphics` und `\lstinputlisting` aus den Unterverzeichnissen `fig`, `pdf` und `code` automatisch eingebunden. Weitere benötigte MP-Dateien werden aus `list` und `defs` mit `\lstinputlisting` geholt.

Gliederung des Buches

Entsprechend meiner Absicht, ein Buch für den praktischen Gebrauch von MP zu schreiben, beginnt es mit einem Schnelleinstieg für den ungeduldigen Leser, um die grundsätzlichen Arbeitsschritte auszuprobieren und die Freude am ersten eigenen Bild zu erleben. Mit diesem Wissen kann man in Kapitel 3 bereits einige interessante Kurven zeichnen. Etwas grundsätzlichere Überlegungen über Technisches Zeichnen, Maße, Normen und Typografie folgen in Kapitel 4. Für das praktische Arbeiten mit MP erweitern wir in Kapitel 5 die eingangs gezeigten Arbeitsschritte zu einer generellen Arbeitsanweisung

(Workflow) und stellen den schematischen Aufbau der MP-Quelldateien als Musterdateien (Templates) zusammen. Dies ermöglicht die sichere Verwendung von TEX/LATEX-Text zur Beschriftung unserer Zeichnungen in der richtigen Schriftgröße und mit dem gewünschten Schriftfont einheitlich für alle Bilder. Die Einzelheiten werden aber erst in Kapitel 10 und am Ende von Kapitel 13 voll verständlich werden. Bis dahin sind der vorgestellte Workflow und die Templates einfach als Rezepte zu betrachten, um überhaupt erst mal solide arbeitsfähig zu werden. Von besonderer Bedeutung ist die Erzeugung von systemunabhängigen, selbständigen (*stand-alone*) eps- und pdf-Dateien für die Bilder mit korrekten Abmessungen (Boundingbox). Das Shell-Skript fasst alle Bearbeitungsschritte zusammen und erleichtert damit das Arbeiten erheblich.

Die folgenden sieben Kapitel behandeln die Eigenschaften der Programmiersprache MP und sind dem syntaktischen Aufbau der Sprache gewidmet. Schwere Kost, die nicht zum sofortigen Verzehr gedacht ist. Beim ersten Lesen sollte man sich nicht entmutigen lassen und sich einfach mal mit den Begriffen und Befehlen vertraut machen, damit man sich bei späterem Bedarf an das eine oder andere erinnert und dann gezielt das Benötigte nachlesen kann. Vielleicht können die eingestreuten Beispiele das Problem ein wenig mildern. Im einzelnen geht es in Kapitel 6 um die Bildung von Variablennamen, die in MP die gleiche Rolle spielen wie in der Algebra. In Kapitel 7 werden dann die zehn verschiedenen Variablentypen und ihre Verknüpfungen besprochen, die den Kern der Ausdrucksmöglichkeiten von MP darstellen. Den grafischen Grundelementen Punkt, Linie, Teilbild, Farbe entsprechen die Variablentypen `pair`, `path`, `picture`, `color` sowie zum Zeichnen `pen` als Feder. Zahlenwerte und Abmessungen wie Länge und Breite entsprechen dem Typ `numeric`. Zur Platzierung von Bildteilen durch Skalierung, Verschiebung und Rotation dienen Transformationen vom Typ `transform`. Die restlichen beiden Typen `boolean` und `string` gibt es in jeder Programmiersprache zur Formulierung von Bedingungen und zur Verarbeitung von Zeichenketten.

MP kennt wie jede Programmiersprache Kontrollstrukturen für Wiederholungen und Verzweigungen zur Steuerung des Programmflusses (Kapitel 8). Die wichtigsten mathematischen Rechenoperationen und elementaren Funktionen, die Rundungs- und Wandeloperationen sind in Kapitel 9 tabellarisch zusammengestellt und erläutert. Kapitel 10 behandelt die Beschriftung unserer Zeichnungen mit dem `abel`-Befehl, einschließlich der Besonderheit der dynamischen Beschriftung. Die Thematik wird am Ende von Kapitel 13 nochmals zusammenfassend aufgegriffen. Der Datenaustausch mit der Außenwelt, also das Lesen und Schreiben von Daten von und auf externe Dateien, Bildschirm und Tastatur wird in Kapitel 11 beschrieben. Den Abschluss dieses Buchteils bildet Kapitel 12, das die Definition und Verwendung von Unterprogrammen (Makros) behandelt, die eine sehr leistungsfähige Erweiterung der Funktionalität von MP ermöglichen.

Mit Kapitel 13 beginnt der praktische Teil des Buches. Da die meisten Zeichnungen, die wir benötigen werden, Diagramme sind, welche Funktionsverläufe darstellen, spielt das Paket `graph.mp` von John D. Hobby eine zentrale Rolle. Es ist in der Lage, mit einer einzigen Anweisung die Daten aus einer externen Datendatei einzulesen und daraus völlig selbständig den Kurvenverlauf in ein komplettes Diagramm mit Skalierung, Achsen, Markierungen (Ticks), Bezifferung und Beschriftung zu zeichnen. Wir werden daher dieses leistungsfähige Paket auf zahlreiche Beispiele anwenden, um die Anpassung an

unterschiedliche Forderungen zu demonstrieren. Das Paket graph lädt und verwendet auch das Paket format, das die Formatierung von Zahlenwerten in Fest- und Gleitpunktdarstellung mit gewünschter Stellenzahl ermöglicht. Der entsprechende Befehl `format` beeinflusst also auch die Beschriftung unserer Zeichnungen, deshalb erfolgt erst hier die vollständige Beschreibung des Beschriftungsproblems.

In Kapitel 14 lernen wir von den großen Meistern ihres Fachs, wie man qualitativ hochwertige Diagramme entwirft und welche goldenen Regeln dabei beachtet werden sollten.

Im folgenden Kapitel konzentrieren wir uns auf weitere häufig verwendete Formen von Diagrammen wie Block-, Struktur-, Fluss- und Balkendiagramme, die Darstellung statistischer Daten durch Box-Plots, Scatter-Plots, Regressionsgeraden und ROC-Kurven. Weiterhin wird gezeigt, wie man externe Bilder im eps-Format in MP einfügt, einen Text entlang eines Pfades schreibt und Pfeilspitzen auf Pfade setzt sowie weitere praktische Konstrukte wie Schraffuren, Superellipsen, geschweifte Klammern und das Lot auf eine Gerade.

Das Kapitel 16 zeigt an einer etwas umfangreicheren Problemstellung, der perspektivischen Darstellung von dreidimensionalen Objekten, die besondere Leistungsfähigkeit von MP, auch mathematisch anspruchsvolle Zusammenhänge zu formulieren und durch Zusammenfassung der einzelnen Schritte in Makros ein komplexes Problem praxisgerecht und kompakt zu lösen. Die Anwendung der Orthogonalprojektion auf die Erdkugel folgt im Wesentlichen der exzellenten Arbeit von Berndt E. Schwerdtfeger. Ein Beispiel zeigt im Rahmen der konformen Abbildung die dreidimensionale Darstellung des Betrags einer komplexwertigen Funktion als Relief.

Dem Leser verbleibt die Aufgabe, die im Verlauf des Buches entwickelten Makros in das Unterverzeichnis `defs` und die Programme und Hilfsdateien in das Unterverzeichnis `list` zu schreiben, um die Beispiele selbst ausprobieren zu können. Für jedes Beispiel schreibt man den MP-Code ebenfalls in das Unterverzeichnis `list` und bearbeitet es mit einem Shell-Skript, das den Code, die eps- und pdf-Datei des Bildes in den Unterverzeichnissen `code`, `fig` und `pdf` archiviert.