
Einführung in die Programmierung mit LOGO

Juraj Hromkovic

Einführung in die Programmierung mit LOGO

Lehrbuch für Unterricht und
Selbststudium

2. Aufl. 2012

Prof. Juraj Hromkovic
ETH Zürich, Schweiz

ISBN 978-3-8348-1852-2
DOI 10.1007/978-3-8348-2266-6

ISBN 978-3-8348-2266-6 (eBook)

Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über <http://dnb.d-nb.de> abrufbar.

Springer Vieweg

© Vieweg+Teubner Verlag | Springer Fachmedien Wiesbaden 2010, 2012

Das Werk einschließlich aller seiner Teile ist urheberrechtlich geschützt. Jede Verwertung, die nicht ausdrücklich vom Urheberrechtsgesetz zugelassen ist, bedarf der vorherigen Zustimmung des Verlags. Das gilt insbesondere für Vervielfältigungen, Bearbeitungen, Übersetzungen, Mikroverfilmungen und die Einspeicherung und Verarbeitung in elektronischen Systemen.

Die Wiedergabe von Gebrauchsnamen, Handelsnamen, Warenbezeichnungen usw. in diesem Werk berechtigt auch ohne besondere Kennzeichnung nicht zu der Annahme, dass solche Namen im Sinne der Warenzeichen- und Markenschutz-Gesetzgebung als frei zu betrachten wären und daher von jedermann benutzt werden dürften.

Gedruckt auf säurefreiem und chlorfrei gebleichtem Papier

Springer Vieweg ist eine Marke von Springer DE.

Springer DE ist Teil der Fachverlagsgruppe Springer Science+Business Media
www.springer-vieweg.de

Vorwort

Dieses Lehrbuch bietet eine Einführung in das Programmieren. Mit seinen ersten sieben Lektionen ermöglicht es sogar einen Einstieg ab dem vierten Schuljahr und mit seinen letzten Lektionen ist es mit dem Mathematikunterricht in den letzten Gymnasialklassen verzahnt. Dabei geht es nicht darum, eine höhere Programmiersprache zu erlernen, sondern vielmehr darum, die fundamentalen Konzepte des systematischen Programmierens und ein tieferes Verständnis zu erwerben. Die notwendige Programmierumgebung ist kostenlos verfügbar.

Hilfreiche Unterstützung anderer hat zur Entstehung dieses Lehrbuches wesentlich beigetragen. Besonderer Dank gilt Karin Freiermuth, Roman Gächter, Stephan Gerhard, Fadri Grünenfelder, Lucia Keller, Dennis Komm, Andre Macejko, Jela Sherlak, Ute Sprock, Andreas Sprock und Björn Steffen für sorgfältiges Korrekturlesen, zahlreiche Verbesserungsvorschläge und umfangreiche Hilfe bei der Umsetzung des Skriptes in $\text{\LaTeX}$. Für die Sprachkorrekturen bedanke ich mich herzlich bei Frau Regina Lauterschläger. Ich möchte mich sehr bei Karin Freiermuth, Barbara Keller, Lucia Keller und Björn Steffen dafür bedanken, dass sie mich mit viel Enthusiasmus beim Testen der Unterrichtsunterlagen in der schulischen Praxis begleitet haben oder einige Lektionen selbstständig unterrichtet haben.

Genauso herzlich danke ich den Lehrpersonen Pater Paul (Hermann-Josef-Kolleg, Steinfeld), Uwe Bettscheider (INDA Gymnasium Aachen), Hansruedi Müller (Schweizerische Alpine Mittelschule Davos), Yves Gärtner, Ueli Marty (Kantonsschule Reussbühl), Meike Akveld, Stefan Meier, Pietro Gilardi (Mathematisch-naturwissenschaftliches Gymnasium Rämibühl, Zürich), Harald Pierhöfer (Kantonsschule Limattal, Urdorf), Michael Weiss (Gymnasium Münchenstein), Paul Eller (Primarschule Attinghausen), Pascal Lütscher (Schulhaus Caguils), Sonja Diggelmann (Primarschule Tuma Platta), Johannes Flury, Levi Flepp und Bernhard Matter (Pädagogische Hochschule Graubünden), Josef Vogt (Kantonsschule Sargans), Hanspeter Buchli (Schule Saas) und Jeannette Brunner (Primarschule Emmenbrücke), die es uns ermöglicht haben, in einigen Klassen kürzere oder längere Unterrichtssequenzen zu testen oder sie sogar selbst getestet haben und uns ihre Erfahrungen mitgeteilt haben. Ein besonderer Dank geht auch an die

Schulleitungen der Schulen, die uns für das Testen unserer Module die Türen geöffnet haben.

Für die hervorragende Zusammenarbeit und die Geduld mit einem immer eigensinniger werdenden Professor bedanke ich mich herzlich bei Frau Kerstin Hoffmann und Herr Ulrich Sandten vom Verlag Springer Vieweg.

Ich wünsche allen Leserinnen und Lesern beim Lernen mit diesem Buch so viel Vergnügen, wie wir selbst beim Unterrichten der vorliegenden Lektionen empfunden haben.

Zürich, im Februar 2012

Juraj Hromkovič

Inhaltsverzeichnis

1	Programme als Folge von Befehlen	15
2	Einfache Schleifen mit dem Befehl repeat	33
3	Programme benennen und aufrufen	53
4	Zeichnen von Kreisen und regelmäßigen Vielecken	71
5	Bewegte Bilder und Zeichentrickfilme	83
6	Programme mit Parametern	97
7	Übergabe von Parameterwerten an Unterprogramme	113
8	Optimierung der Programmlänge und der Berechnungskomplexität	131
9	Das Konzept von Variablen und der Befehl make	147
10	Lokale und globale Variablen	169
11	Verzweigungen von Programmen und while -Schleifen	183
12	Integrierter LOGO- und Mathematikunterricht: Geometrie und Gleichungen	203
13	Rekursion	217
14	Integrierter LOGO- und Mathematikunterricht: Trigonometrie	245
15	Integrierter LOGO- und Mathematikunterricht: Vektorgeometrie	257

Einleitung

Programmieren in LOGO

Warum unterrichten wir Programmieren?

Programmieren gehört zum Handwerkszeug eines jeden Informatikers, auch wenn das Programmieren alleine noch keinen Informatiker ausmacht. Vor ungefähr zwanzig Jahren war Programmieren in Ländern mit Informatikunterricht an den Mittelschulen ein hauptsächlicher Bestandteil des Curriculums. Dann folgte eines der unglücklichsten Eigentore, die die Informatiker sich je geschossen haben. Einige von ihnen haben begonnen, die Wichtigkeit der Informatik und deren Unterricht damit zu begründen, dass nun fast jeder einen Rechner habe oder bald haben werde. Man müsse deswegen mittels Informatikunterricht den kompetenten Umgang mit dem Rechner inklusive aktueller Software lehren. Dies ist eine ähnlich gelungene Begründung wie die eines Maschinenbauers, welcher den Unterricht seines Faches an Gymnasien damit rechtfertigen würde, dass fast jedermann ein Fahrzeug besitze und man deswegen allen die Chance geben müsse, damit kompetent umgehen zu lernen.

Die unmittelbare Folge dieser „*Propaganda*“ in einigen Ländern war der Ausbau oder Umbau des bestehenden Informatikunterrichts zu einer billigen Ausbildung zum Computerführerschein. Die Vermittlung von Wissen und informatischer Grundfertigkeit wurde durch das Erlernen des Umgangs mit kurzlebigen und größtenteils mangelhaften Softwaresystemen ersetzt. Es dauerte dann auch nur wenige Jahre, bis die Bildungspolitiker und Schulen erkannt haben, dass eine solche Informatik weder Substanz noch Nachhaltigkeit besitzt, und man für einen Computerführerschein kein eigenständiges Fach Informatik braucht. Und so hat man dann umgehend „*das Kind mit dem Bade ausgeschüttet*“. Wir haben heute in den meisten Gebieten des deutschsprachigen Raumes keinen eigentlichen Informatikunterricht mehr in den Mittelschulen.

Zwar wurde das Problem inzwischen erkannt, doch der Informatikunterricht wird an den Folgen der billigen „*Informatik-Propaganda*“ noch viele Jahre zu leiden haben, nicht nur weil es schwierig ist, in kurzer Zeit und ohne entsprechend ausgebildete Lehrpersonen von einem schlecht eingeführten Unterricht zu einem qualitativ hochwertigen Unterricht zu wechseln, sondern auch, weil die Informatik

in der Öffentlichkeit ein falsches Image erhalten hat. Informatiker sind diejenigen, die mit dem Rechner gut umgehen können, das heißt alle Tricks kennen, um jemandem bei den allgegenwärtigen Problemen mit unzulänglicher Software zu helfen. Wir sollten dieses Bild mit der Wertigkeit von Fächern wie Mathematik, Physik und anderen Gymnasialfächern vergleichen. Auch wenn diese Fächer nicht bei jedermann beliebt sind, wird ihnen niemand die Substanz absprechen wollen.

Im Gegensatz dazu hören wir von guten Gymnasialschülerinnen und -schülern oft, dass ihnen die Informatik zwar Spaß macht, dass sie zum Studieren aber „zu leicht“ sei: „Das kann man sich nebenbei aneignen.“

Sie wollen ein Fach studieren, welches eine echte Herausforderung darstellt. In dem, was sie bisher im sogenannten Informatikunterricht gesehen haben, sehen sie keine Tiefe oder Substanz, für deren Beherrschung man sich begeistern lassen kann.

Andererseits wissen wir, dass sich die Informatik inzwischen gewaltig entwickelt hat, so dass dank ihr in vielen Gebieten der Grundlagenforschung sowie der angewandten technischen Disziplinen wesentliche Fortschritte erzielt werden konnten. Sowohl die Anzahl der Anwendungen als auch die der Forschungsrichtungen der Informatik ist in den letzten Jahren so stark gewachsen, dass es sehr schwierig geworden ist, ein einheitliches, klares Bild der Informatik zu vermitteln, ein Bild einer Disziplin, die in sich selbst die mathematisch-naturwissenschaftliche Denkweise mit der konstruktiven Arbeitsweise eines Ingenieurs der technischen Wissenschaften verbindet. Die Verbindung dieser unterschiedlichen Denkweisen und Wissenschaftssprachen in einem einzigen Fach ist aber gerade die Stärke des Informatikstudiums.

Die Hauptfrage ist nun, wie man diese vielen, sich dynamisch entwickelnden Informatikgebiete, -themen und -aspekte in ein Curriculum fürs Gymnasium abbilden kann. Da divergieren die Meinungen und Präferenzen der Informatiker so stark, dass man nur sehr schwer einen Konsens erreichen kann.

Warum sehen wir in dieser, von allerlei widersprüchlichen Meinungen geprägten Situation das Programmieren als einen unbestrittenen und zentralen Teil der Informatikausbildung an? Dafür gibt es mehrere Gründe: Auch andere Gymnasialfächer stehen vor keiner einfachen Wahl. Und wir können einiges von ihnen lernen, zum Beispiel, dass wir die historische Entwicklung verfolgen sollten, anstatt uns ausschließlich auf die Vermittlung der neuesten Entwicklungen zu konzentrieren. In Fächern wie Mathematik oder Physik ist der Versuch, um jeden Preis neueste Entdeckungen zu vermitteln, didaktisch geradezu selbstvernichtend.

Wenn wir als Informatiker unserer eigenen Disziplin also eine ähnliche Tiefe zuschreiben, dürfen wir uns nicht auf diesen Irrweg einlassen. Wir müssen bodenständig bleiben und wie in anderen Fächern mit der Begriffsbildung und

Grundkonzepten beginnen. Die historisch wichtigsten Begriffe, welche die Informatik zur selbständigen Disziplin gemacht haben, sind die Begriffe „**Algorithmus**“ und „**Programm**“. Und wo könnte man die Bedeutung dieser Begriffe besser vermitteln als beim Programmieren? Dabei verstehen wir das Programmieren nicht nur als eine für den Rechner verständliche Umsetzung bekannter Methoden zur Lösung gegebener Probleme, sondern vielmehr als die Suche nach konstruktiven Lösungswegen zu einer gegebenen Aufgabenstellung. Wir fördern dabei einerseits die Entwicklung des algorithmischen, lösungsorientierten Denkens und stehen damit in Beziehung zum Unterricht der Mathematik, während wir andererseits mit dem Rechner zu „kommunizieren“ lernen.

Programme zu schreiben bedeutet eine einfache und sehr systematisch aufgebaute Sprache, genannt Programmiersprache, zu verwenden. Die Besonderheit der Programmiersprachen ist die Notwendigkeit, sich korrekt, exakt und eindeutig auszudrücken, weil der „Dialogpartner“ unfähig ist zu improvisieren. Wenn absolute Präzision und Klarheit in der Formulierung der Anweisungen unabdingbare Voraussetzung für die unmissverständliche Erklärung der Lösungswege sind, so dass sie sogar eine Maschine ohne Intellekt verstehen und umsetzen kann, fördert dies die Entwicklung der Kommunikationsfähigkeit enorm.

Ein weiterer Grund für die zentrale Bedeutung des Programmierunterrichts liegt in der Verbindung zwischen der logisch-mathematischen Denkweise und der konstruktiven Denkweise der Entwickler in den technischen Wissenschaften. Die Problemspezifikation, die Suche nach einem Lösungsweg sowie die formale Ausdrucksweise zur Beschreibung einer gefundenen Lösungsmethode sind stark mit der Nutzung der Mathematik sowohl als Sprache als auch als Methode verbunden. Ein wesentlicher Lerneffekt bei der Entwicklung komplexer Programme ist die „modulare“ Vorgehensweise. Die Modularität ist typisch für Ingenieurwissenschaften. Zuerst baut man einfache Systeme für einfache Aufgabenstellungen, deren korrekte Funktionalität leicht zu überprüfen ist. Diese einfachen Systeme („Module“) verwendet man als (Grund-) Bausteine zum Bau von komplexeren Systemen. Diese komplexen Systeme kann man selbst wieder als Module (Bausteine) verwenden, aus denen man noch komplexere Systeme bauen kann, usw. Programmieren ist also ein ideales Instrument zum systematischen Unterricht in der modularen Vorgehensweise beim Entwurf komplexer Systeme aller Arten.

Schließlich bietet Programmieren einen sinnvollen Einstieg in die Welt der weiteren grundlegenden Begriffe der Informatik, wie Verifikation, Berechnungskomplexität (Rechenaufwand) und Determiniertheit. Wenn man lernt, wie Programme nach unterschiedlichen Kriterien wie Effizienz, Länge, Verständlichkeit, modulare Struktur, Benutzerfreundlichkeit oder „Kompatibilität“ beurteilt werden können, versteht man auch, dass kein bestehendes System vollkommen ist. Dies führt

zur Fähigkeit, Produkte kritisch zu durchleuchten und zu beurteilen sowie über Verbesserungen nach ausgesuchten Kriterien nachzudenken.

Zusammenfassend trägt der Programmierunterricht auf vielen unterschiedlichen Ebenen zur Wissensvermittlung und Bildung bei. Neben der Fertigkeit, in bestimmten Programmiersprachen zu programmieren, erwerben die Schüler in Projekten die Fähigkeit, die Denkweisen der Theorie und der Praxis miteinander zu verbinden und systematisch, konstruktiv und interdisziplinär zu arbeiten. Indem sie den ganzen Weg von der Idee bis zum fertigen Produkt selbst miterleben, entwickeln sie eine fundierte Haltung zu Entwicklungsprozessen im Allgemeinen. Die Verbindung neuer Ideen mit der selbständigen Überprüfung im Hinblick auf die Umsetzbarkeit bereichert die Schule noch auf eine andere Art und Weise: Es gibt kein anderes Themengebiet der Informatik, dessen Lernprozess derart viele grundlegende Konzepte integriert.

Unser Programmiervorkurs in LOGO, oder Programmieren in 10 Minuten

Beim Programmierunterricht steht das Erlernen der Programmierkonzepte im Vordergrund und das Meistern einer konkreten höheren Programmiersprache muss als zweitrangig gesehen werden. Hier empfehlen wir deswegen, mit LOGO anzufangen und dann zum geeigneten Zeitpunkt zu einer höheren pascalartigen Sprache zu wechseln. Die Zeit, die man am Anfang mit LOGO verbringt, wird später im Unterricht einer höheren Programmiersprache mehr als zurückgezahlt.

Warum gerade LOGO für den Einstieg in die Programmierung? LOGO ist eine Programmiersprache, die aus rein fachdidaktischen Gründen für den Unterricht der Programmierung entwickelt wurde. Aus didaktischer Sicht ist sie genial und uns ist nach vielen Jahren ihrer Existenz noch keine annähernd vergleichbare Konkurrenz bekannt. Sie reduziert die Syntax so stark, dass man mit ihr praktisch nicht belastet ist. Somit können Anfänger schon nach zehn Minuten ihre ersten eigenen Programme schreiben und im Test laufen lassen. Programmierer können Programme zuerst Befehl um Befehl schreiben und einzeln anschauen, ob die jeweiligen Befehle das Gewünschte verursachen. Sie können aber auch zuerst komplette Programme oder Programmteile schreiben und sie dann während eines Durchlaufs überprüfen. Die Möglichkeit, sich am Anfang auf die Entwicklung von Programmen zur Zeichnung von Bildern zu konzentrieren, macht den Lernprozess sehr anschaulich, motivierend und dadurch attraktiver.

Eine hohe Interaktion und gegenseitige Befruchtung mit dem Geometrieunterricht ist leicht zu erreichen. Aus Erfahrung lassen sich alle Altersgruppen für das

Programmieren in LOGO von Anfang an leicht begeistern. Das Programmieren ist lösungsorientiert und prägt die Entwicklung des algorithmischen Denkens.

Der Kern des didaktischen Vorgehens ist es, auf gut zugängliche Weise die folgenden Grundkonzepte des Programmierens zu vermitteln:

- Ein Programm als Folge von einfachen Grundbefehlen aus einer kurzen Befehlsliste
- Unterprogramme als Module zum Bau von komplexen Programmen
- Schleifen als Teil des strukturierten Programmierens
- Variablen als Grundkonzept des Programmierens
- Lokale und globale Variablen und Übertragung von Daten zwischen Programmen
- Verzweigung von Programmen und bedingten Befehlen
- Rekursion

Dabei kann man den Unterricht so gestalten, dass man mit nur fünf bis sechs Befehlen startet. Alles, was man braucht, wird aus diesen wenigen Befehlen zusammengesetzt. Damit verstehen die Schülerinnen und Schüler sofort, dass viele Befehle nur Namen für ganze Programme sind, die selbst aus einfachen Befehlen bestehen. Sie lernen dabei, nach Bedarf eigene neue Befehle zu programmieren, zu benennen und dann einzusetzen. Wegen der Syntax und der großen Vielfalt an komplexen Befehlen bei höheren Programmiersprachen nimmt die Vermittlung dieser Konzepte einen unvergleichbar höheren Aufwand in Anspruch als bei LOGO.

Die Programmierumgebung in LOGO

Für die Herstellung dieses Moduls haben wir die kommerzielle Programmiersprache SUPERLOGO und die freie Software XLOGO verwendet. Beide zeichnen sich dadurch aus, dass ihre Umgebungen so einfach aufgebaut sind, dass man praktisch sofort mit dem Programmieren beginnen kann. XLOGO kann unter der Adresse

<http://xlogo.tuxfamily.org>

heruntergeladen werden. Lösungen zu den in diesem Buch gestellten Aufgaben können unter der folgenden Adresse gefunden werden:

<http://www.abz.inf.ethz.ch/logo>

Der Bildschirm von XLOGO sieht vereinfacht aus wie in Abbildung 0.1 dargestellt. In das **Befehlsfenster** schreibt man die Befehle (Rechnerinstruktionen) für die Schildkröte. Im **Befehlsfenster** kann man einen Befehl oder eine Folge von Befehlen, also sogar ein vollständiges Programm, schreiben. Wenn man die Return-Taste drückt, wird der Rechner die ganze Folge der Befehle aus dem **Befehlsfenster** ausführen. Die Umsetzung der Befehle können wir direkt an den Bewegungen der Schildkröte im **Schildkrötenfenster** beobachten. Man kann den Inhalt des **Befehlsfensters** als eine Programmzeile betrachten. Nach ihrer Ausführung wird die komplette Zeile nach unten in das **Fenster der letzten Befehle** übertragen. Damit behalten wir die Übersicht über den bisher geschriebenen Teil unseres Programms. Die Schaltfläche mit der Beschriftung „STOP“ dient zum Anhalten der Programmausführung. Das ist sinnvoll wenn das Programm zu lange, oder sogar unendlich lange läuft. Für die Ziele der ersten beiden Lektionen reicht dieses Wissen über die Programmierungsumgebung aus.

Hinweis für die Lehrperson Bei den Kursen für Anfänger lohnt es sich, zuerst immer nur einen Befehl in eine Zeile zu schreiben, um seine Wirkung genau beobachten zu können. Ausserdem erleichtert diese Vorgehensweise eine selbstständige Entdeckung von Fehlern.

In Lektion 3 lernen wir, Programme zu benennen und abzuspeichern. Wenn man in XLOGO Programme mit Namen bezeichnen will, muss man auf die Schaltfläche „Editor“ klicken. Dann erscheint ein kleines Fenster wie vereinfacht in Abbildung 0.2 dargestellt. In diesem **Editor** sehen wir alle bisher geschriebenen Programme. Hier können wir neue Programme speichern oder die alten Programme editieren

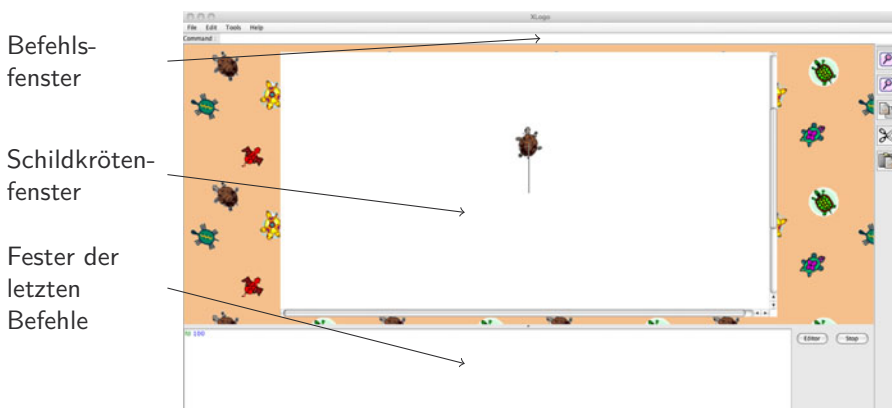


Abbildung 0.1 Der Bildschirm von XLOGO mit dem Schildkrötenfenster und dem Befehlsfenster.

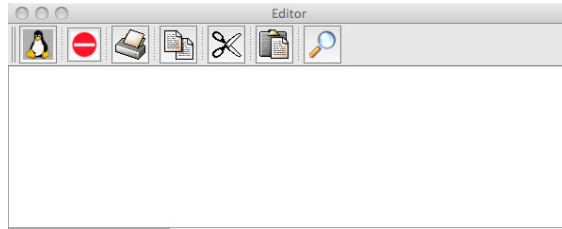


Abbildung 0.2 Der Editor von XLOGO

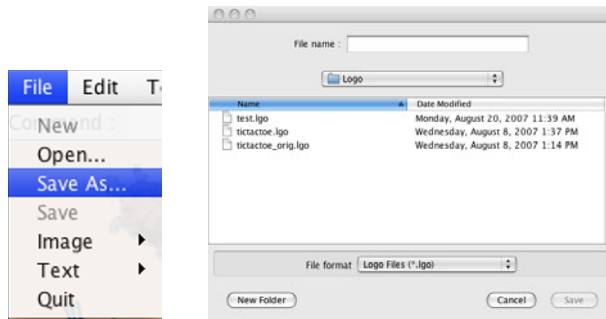


Abbildung 0.3 Links ist das Menü File abgebildet. Wenn man auf „Save As . . . “ klickt, erhält man das Projektfenster rechts.

(verändern). Das Fenster können wir mit einem Mausklick auf die Schaltfläche Pinguin schließen.

Wenn man alle geschriebenen Programme unter einem gewählten Projektnamen abspeichern will (um sie in der nächsten Stunde wieder verwenden zu können), dann klickt man auf das Menü File links oben (Abbildung 0.1). Daraufhin erscheint ein Menü wie in Abbildung 0.3 (links), bei dem man „Save As . . . “ auswählen muss. Danach erscheint das Projektfenster, wie es in Abbildung 0.3 (rechts) gezeigt ist. Wir wählen ein Verzeichnis für die Abspeicherung von LOGO-Projekten und suchen uns einen Namen für das zu speichernde Projekt, den wir in das Textfeld File name eintippen. Mit dem Klick auf „Save“ werden dann die bisher geschriebenen Programme unter dem angegebenen Projektnamen gespeichert.

Die Programmierumgebung SUPERLOGO ist noch etwas benutzerfreundlicher und ermöglicht uns, viele Aktivitäten direkt auf dem Basisbildschirm umzusetzen. Der Bildschirm sieht zu Beginn wie in Abbildung 0.4 aus.

Das Programmierfenster in SUPERLOGO erfüllt die Funktionen vom Befehlsfenster und vom Fenster der letzten Befehle in XLOGO. Die letzte geschriebene Zeile ist die aktuelle und nach dem Drücken von „Return“ werden die dort geschriebenen Befehle ausgeführt. Das Schildkrötenfenster funktioniert genau so wie

bei XLOGO. Die horizontale Linie zwischen dem Schildkrötenfenster und dem Programmierfenster kann man beliebig in der Vertikalen verschieben und so die Proportionen zwischen diesen beiden Fenstern beliebig variieren.

Die Benennung von Programmen werden direkt im Programmierfenster durch den Befehl `to` vorgenommen, wie in Lektion 3 beschrieben. Wenn man bereits geschriebene Programme anschauen oder verändern will, muss man auf die vierte Schaltfläche von links in der oberen Zeile des Bildschirms klicken. Danach erscheint das Fenster aus Abbildung 0.5. Im linken Teil dieses Fensters sind die Namen aller gespeicherter Programme aufgelistet. Durch einen Doppelklick auf einen Namen erscheint das gewünschte Programm im rechten Fenster. Man kann sich beliebig viele Programme gleichzeitig anschauen. Durch einen Doppelklick auf ein Programm aus dem rechten Teil des Fensters öffnet sich ein neues Fenster, in dem man das gewählte Programm editieren (verändern) kann.

Wenn man die bisher geschriebenen Programme unter einem Projektnamen speichern will, dann klickt man auf die zweite Schaltfläche von links in der obersten Zeile. Danach erscheint das Savefenster aus Abbildung 0.6. Jetzt können wir auf den Namen eines schon definierten Projekts in dem größeren Teilfenster links klicken. Dadurch erscheint dieser Name im Textfeld **Projektname**. Mit einem Klick auf „OK“ wird das Projekt unter diesem Namen gespeichert. Wir können aber auch einen völlig neuen Namen für das Projekt wählen, indem wir den Namen direkt in das Textfeld **Projektname** eintippen.

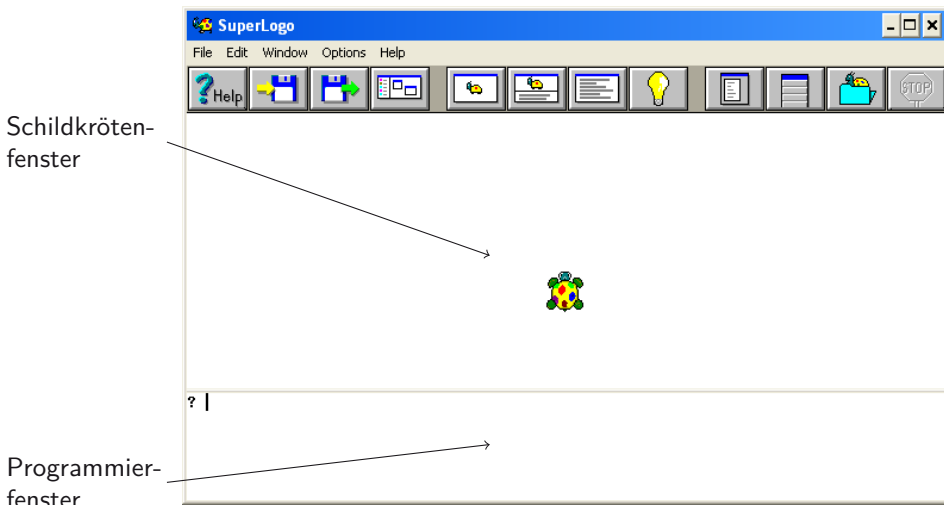


Abbildung 0.4 Die Programmierumgebung SUPERLOGO mit dem Programmierfenster und dem Schildkrötenfenster.

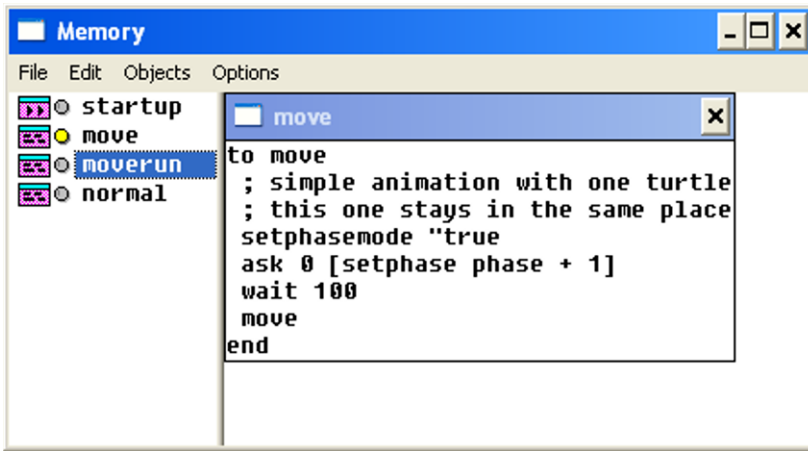


Abbildung 0.5

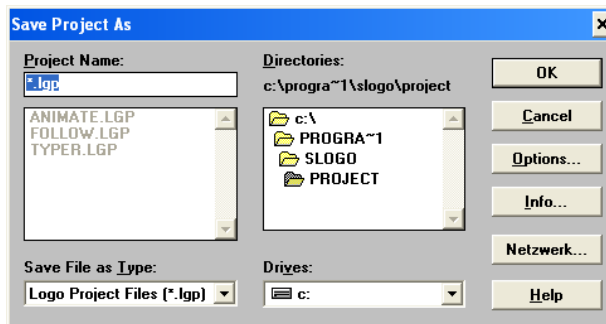


Abbildung 0.6

Wenn wir von Anfang an wissen, in welchem Projekt wir arbeiten wollen, und wissen, dass wir unsere neuen Programme zusätzlich zu den alten in dasselbe Projekt speichern wollen, müssen wir eben dieses Projekt zu Beginn unserer Arbeit laden. Dazu klicken wir auf die dritte Schaltfläche von links in der obersten Zeile und erhalten ein ähnliches Fenster wie in Abbildung 0.7. Danach können wir aus der Liste der bisherigen Projektnamen ein Projekt auswählen und dadurch den Projektnamen in das Textfeld Projektname holen. Mit einem Klick auf „OK“ bestätigen wir unsere Wahl.

Der häufigste Fehler passiert, wenn man am Anfang der Arbeit kein Projekt geladen hat und am Ende unter einem schon existierenden Projektnamen die neuen Programme abspeichert. Bei diesem Vorgehen werden zwar die neu geschriebenen

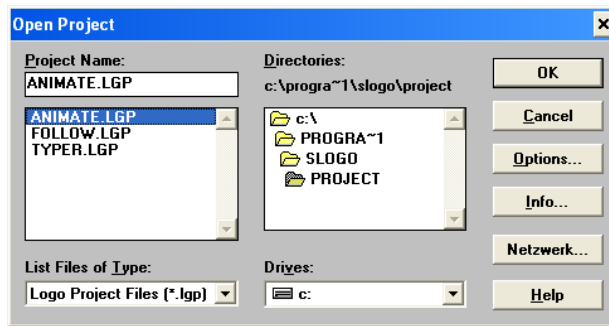


Abbildung 0.7

Programme gespeichert, jedoch werden alle alten Programme, die zuvor in diesem Projekt gespeichert waren, gelöscht.