Deutsches
Forschungszentrum
für Künstliche
Intelligenz GmbH

Document
D-95-02

BETTY

Ein System zur Planung und Generierung informativer Animationssequenzen

Andreas Butz

February 1995

Deutsches Forschungszentrum für Künstliche Intelligenz GmbH

Postfach 20 80
67608 Kaiserslautern, FRG
Tel.: + 49 (631) 205-3211
Fax: + 49 (631) 205-3210

Stuhlsatzenhausweg 3
66123 Saarbrücken, FRG
Tel.: + 49 (681) 302-5252
Fax: + 49 (681) 302-5341

Deutsches Forschungszentrum für Künstliche Intelligenz

The German Research Center for Artificial Intelligence (Deutsches Forschungszentrum für Künstliche Intelligenz, DFKI) with sites in Kaiserslautern and Saarbrücken is a non-profit organization which was founded in 1988. The shareholder companies are Atlas Elektronik, Daimler-Benz, Fraunhofer Gesellschaft, GMD, IBM, Insiders, Mannesmann-Kienzle, SEMA Group, and Siemens. Research projects conducted at the DFKI are funded by the German Ministry for Research and Technology, by the shareholder companies, or by other industrial contracts.

The DFKI conducts application-oriented basic research in the field of artificial intelligence and other related subfields of computer science. The overall goal is to construct *systems with technical knowledge and common sense* which - by using AI methods - implement a problem solution for a selected application area. Currently, there are the following research areas at the DFKI:

- Intelligent Engineering Systems
- Intelligent User Interfaces
- Computer Linguistics
- Programming Systems
- Deduction and Multiagent Systems
- Document Analysis and Office Automation.

The DFKI strives at making its research results available to the scientific community. There exist many contacts to domestic and foreign research institutions, both in academy and industry. The DFKI hosts technology transfer workshops for shareholders and other interested groups in order to inform about the current state of research.

From its beginning, the DFKI has provided an attractive working environment for AI researchers from Germany and from all over the world. The goal is to have a staff of about 100 researchers at the end of the building-up phase.

Dr. Dr. D. Ruland
Director

BETTY

Ein System zur Planung und Generierung informativer Animationssequenzen

Andreas Butz

DFKI-D-95-02

This work has been supported by a grant from The Federal Ministry for Research and Technology (FKZ ITWM-9400).

ISSN 0946-0098

BETTY

Ein System zur Planung und Generierung informativer Animationssequenzen

Andreas Butz

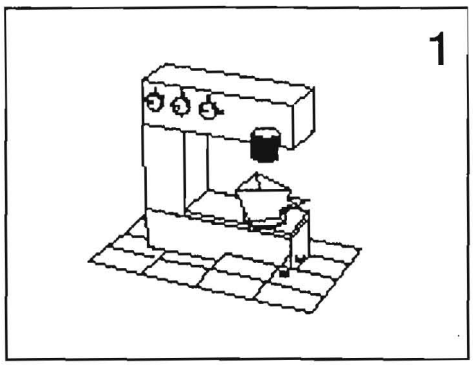
14. Februar 1995

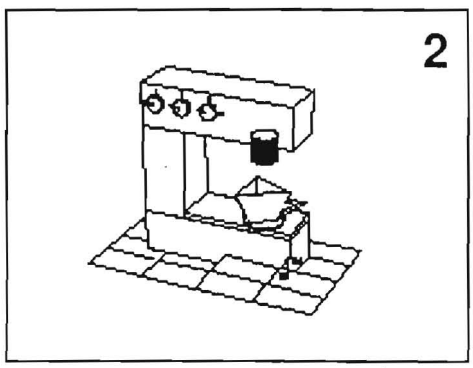
Bei der Präsentation von Informationen am Bildschirm werden klassischerweise die Präsentationsmodi Text und statische Grafik benutzt. Insbesondere am Bildschirm ist es aber möglich, Informationen auch als bewegte Grafiken, sprich Animationen, darzustellen. Die vorliegende Arbeit befaßt sich mit der automatischen Erzeugung von 3D-Animationssequenzen aus rein inhaltlichen Beschreibungen. Diese Animationssequenzen sind Bestandteil einer multimodalen Präsentation und drücken bestimmte Vorgänge und/oder Sachverhalte aus.

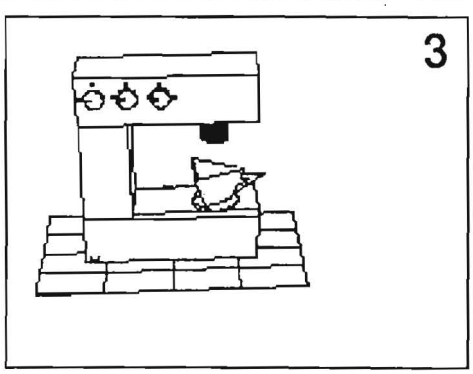
Ein Vorteil des Modus Animation liegt darin, daß die Dimension Zeit (Reihenfolge, Dauer) direkt mit wiedergegeben wird, ein weiterer darin, daß es in bewegten Bildern wesentlich besser als in statischen Grafiken gelingt, scheinbar die Grenze zur dritten Raumdimension zu überspringen.

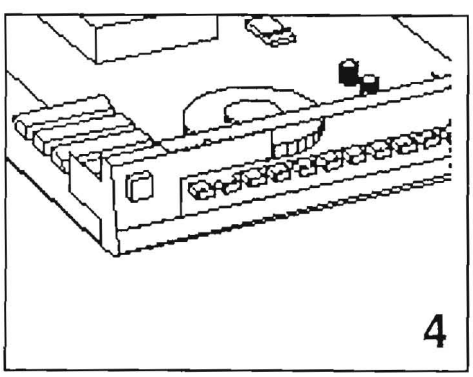
Das Problem, eine Computeranimation zu gestalten, wurde in dieser Arbeit als Planungsproblem aufgefaßt, und das Ergebnis ist ein Animationsplaner, der ausgehend von einem Visualisierungsziel eine Animationssequenz mit Objektbewegungen und Kameraführung plant, die anschließend von einem kommerziellen Animationssystem berechnet und abgespielt wird.

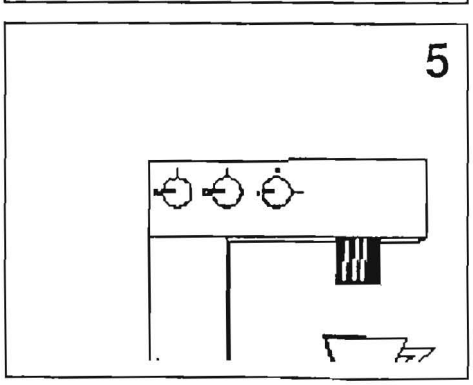
BETTY ist Bestandteil der multimodalen wissensbasierten Benutzerschnittstelle WIP (Wissensbasierte Informations-Präsentation, [Wa93]), die automatisch aus geometrischen Modellen technischer Geräte und symbolischen Repräsentationen der mit ihnen ausführbaren Handlungen an verschiedene Situationen angepaßte Bedienungsanleitungen generiert.


1

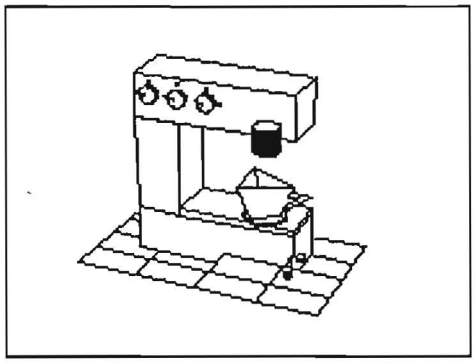

2

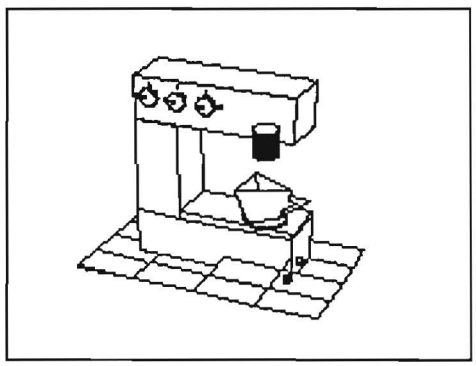

3

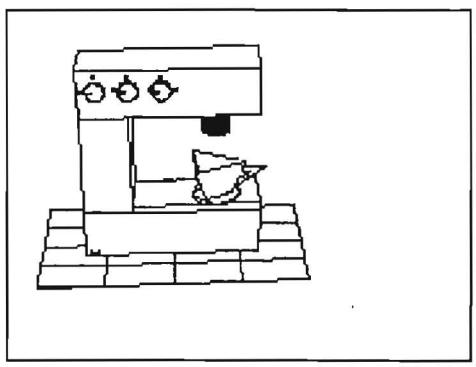

4

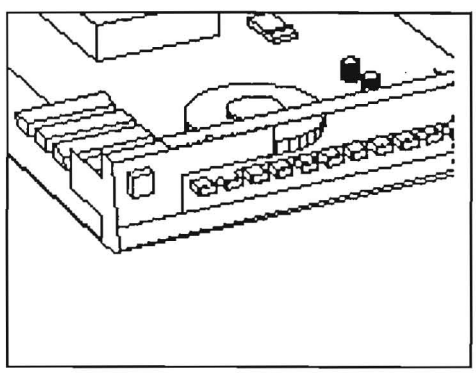

5

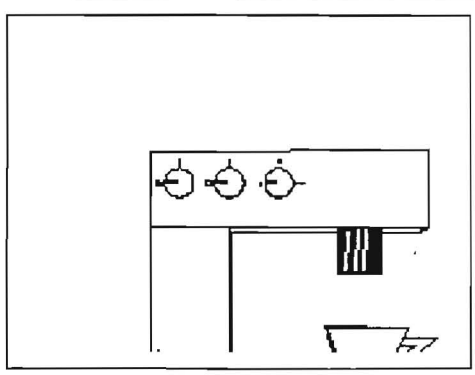
Inhaltsverzeichnis

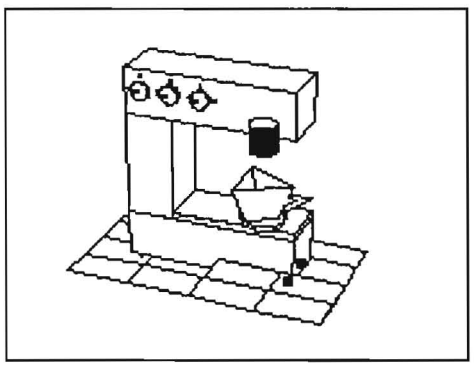


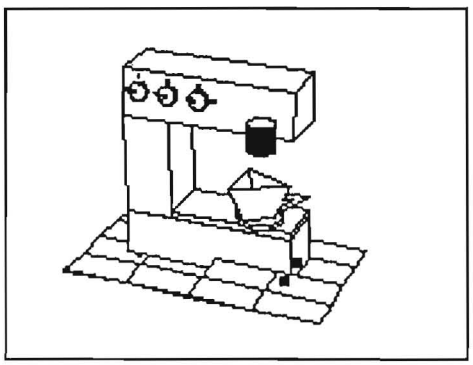


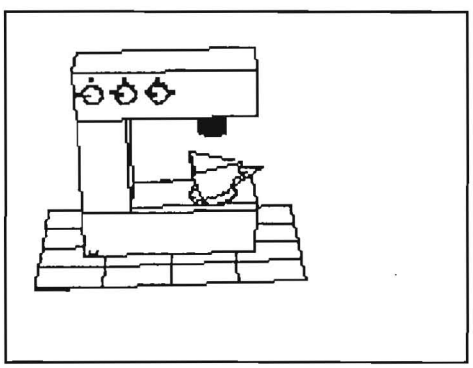


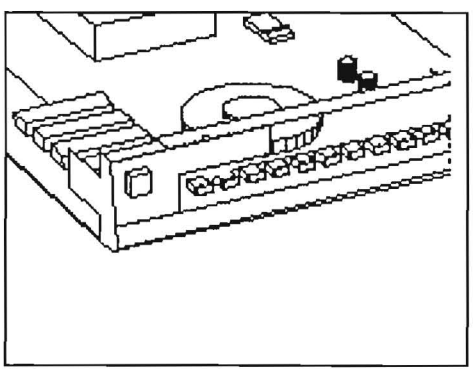


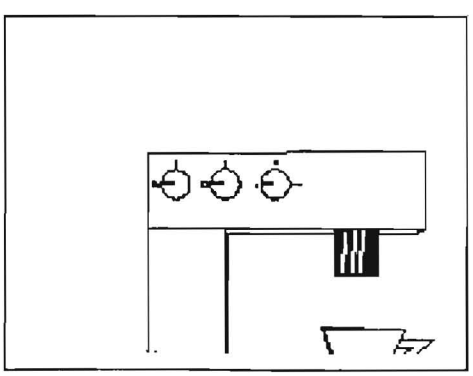




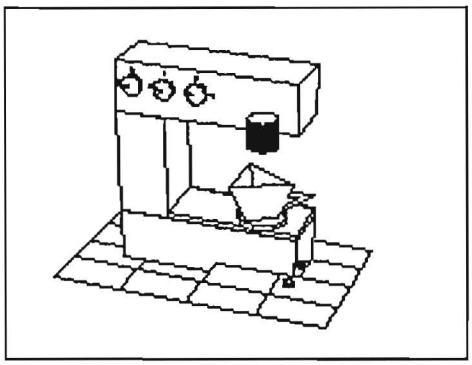


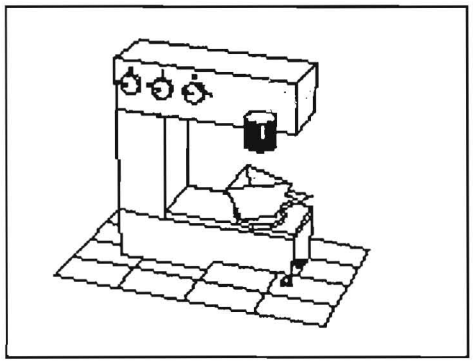


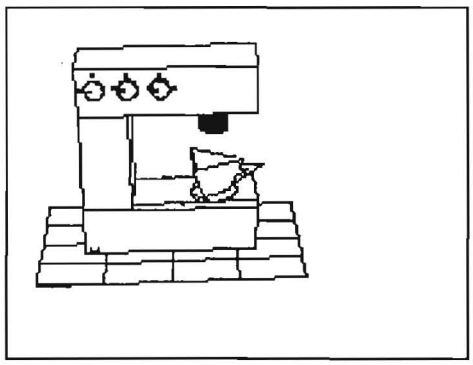


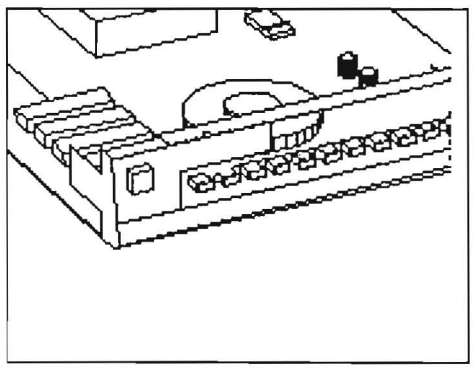


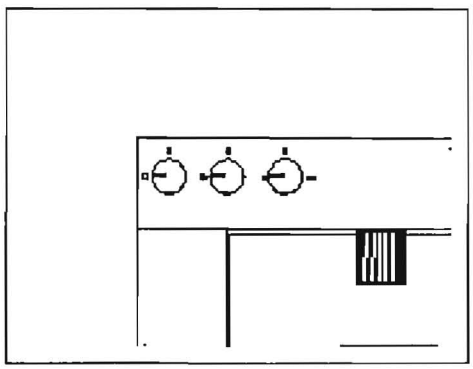
Abbildungsverzeichnis











Vorwort zum DFKI-Dokument

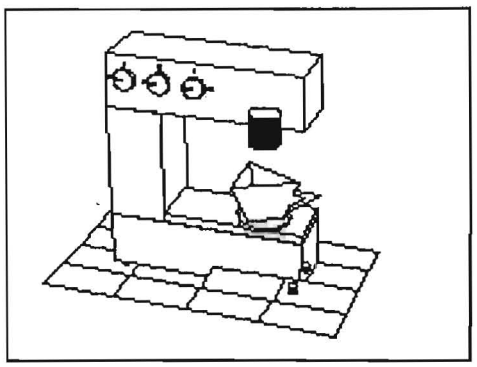
Die vorliegende Arbeit wurde als Diplomarbeit in Informatik im September 1994 dem Prüfungsamt der Technischen Fakultät der Universität des Saarlandes in Saarbrücken vorgelegt. Aufgrund bestehender Nachfrage liegt sie nun als DFKI-Dokument vor und ist somit über die Bibliothek des DFKI jedem zugänglich. Der Text selbst wurde unverändert übernommen, lediglich das Druckformat wurde soweit möglich an die bestehenden DFKI-Dokumente angeglichen.

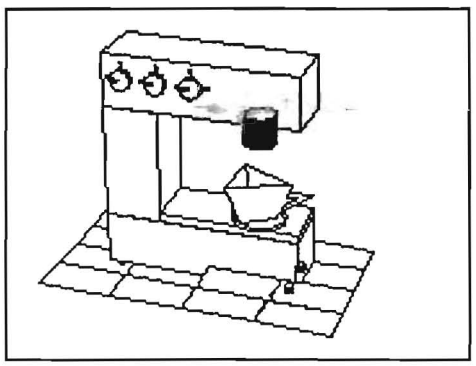
Danksagung

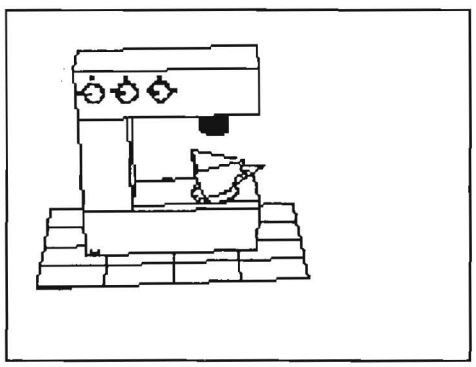
Danken möchte ich meinem Betreuer Prof. Dr. Wolfgang Wahlster für die Unterstützung meiner Arbeit durch aktuelle Literaturhinweise, Anregungen und konstruktive Kritik, sowie Frau Elisabeth André für Diskussionen, Anregungen und weitere Literatur. Die menschliche Unterstützung durch Tatjana Klajić, vor allem in der Endphase der Arbeit, sowie die menschliche und finanzielle Unterstützung durch meine Eltern erleichterten die Ausführung ebenfalls beträchtlich.

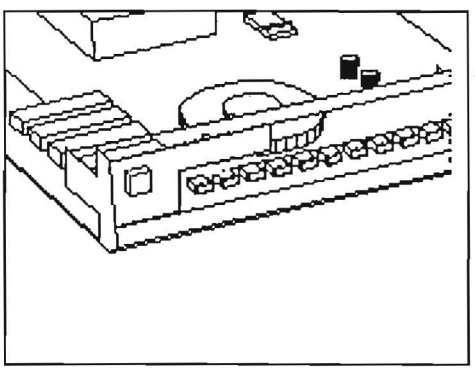
An dieser Stelle möchte ich auch vermerken, daß ohne ein erhebliches Repertoire an bereits vorhandenen Funktionen zum automatischen Grafik-Design, die von verschiedenen Mitstudenten, Mitarbeitern und ehemaligen Mitarbeitern des DFKI stammen, eine Realisierung der Animationskomponente BETTY im vorliegenden Umfang wesentlich erschwert gewesen wäre und es in der vorhandenen Zeit und im Rahmen einer Diplomarbeit wohl kaum zu einem so gut funktionierenden Gesamtsystem gekommen wäre.

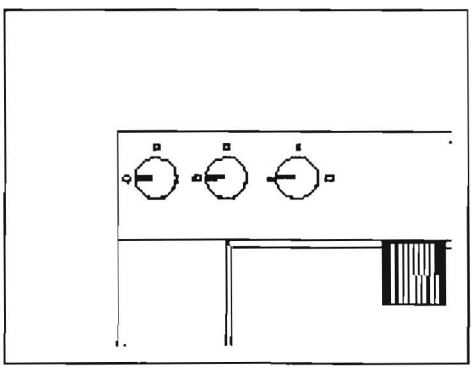
Schließlich läßt sich eine Diplomarbeit in praktischer Informatik auch nur durchführen, solange der störungsfreie Betrieb der benötigten Rechner gewährleistet ist. Allen hierfür verantwortlichen Personen gilt mein ausdrücklicher Dank.











1 Einleitung

1.1 Ziel der Arbeit

Die Zielsetzung dieser Arbeit besteht in erster Linie darin, ein System zu schaffen, das Animationssequenzen alleine aufgrund eines Präsentationszieles, einer *inhaltlichen* Spezifikation, generiert. Hierbei soll die Gestaltung der Animation, die *formale* Spezifikation also, fast vollständig automatisiert werden.

Als Vorgaben werden lediglich die Modelle der animierten Objekte sowie ihrer Bewegungen benutzt. In die Beschreibung solcher Objekt- und Bewegungsmodelle wurde bereits einiger Forschungsaufwand investiert (siehe [Fo90, BBZ90, Br94]) und sie soll in dieser Arbeit nicht näher untersucht werden. In der Tat liegt BETTY sogar ein recht einfaches Objektmodell (wire frames) und Bewegungsmodell (splines) zugrunde, das jedoch zur Untersuchung des prinzipiellen Problems der Animationsplanung die nötige Flexibilität bietet, ohne selbst neue Probleme aufzuwerfen.

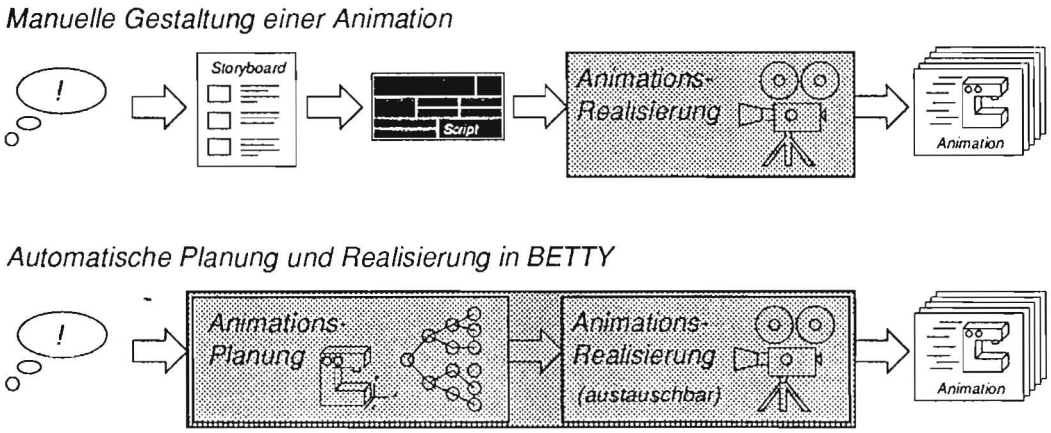
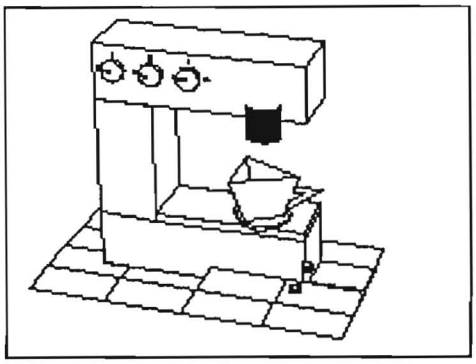
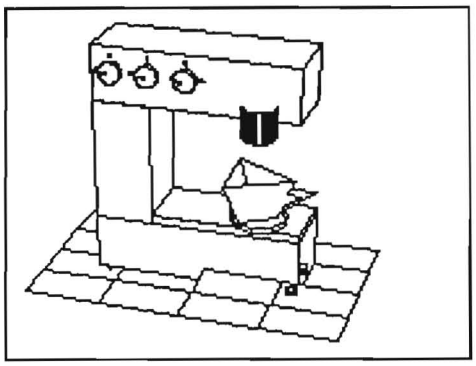
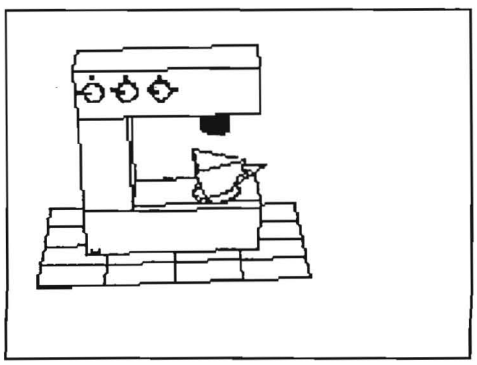


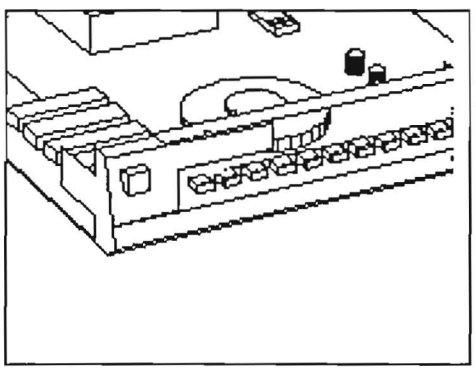
Abbildung 1: Automatisierung der gesamten Animationsgestaltung

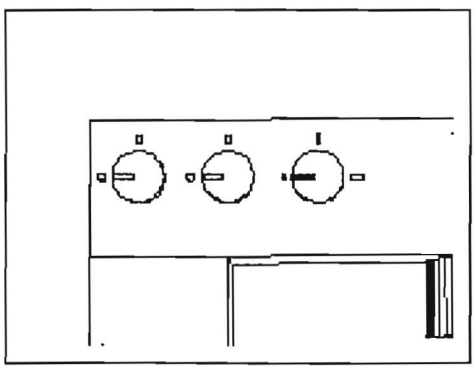
Während herkömmliche Animationen normalerweise vollständig manuell spezifiziert werden müssen, ist es das Ziel dieser Arbeit, die komplette formale Spezifikation einer Animationssequenz aus ihrer inhaltlichen Beschreibung abzuleiten, wozu die gesamte Kameraführung sowie ihre zeitliche und räumliche Koordination mit den gezeigten Objektbewegungen gehört.











Treibt man diese Vorstellung ins Extrem, so müßte mit einem solchen System beispielsweise aus einem Roman (bzw. seiner inhaltlichen Beschreibung) nach vorheriger Modellierung der Schauspieler und der Schauplätze der zugehörige Film zu erzeugen sein. Diese offensichtliche Übertreibung macht aber auch gleichzeitig deutlich, was das System nicht leisten kann:

Die künstlerische Gestaltung von „guten"Animationssequenzen ist wie jede Gestaltungsaufgabe ein kreativer Akt und als solcher nicht oder nur sehr rudimentär in Regeln zu fassen. Die von BETTY erzeugten Animationssequenzen dienen dem Zweck, eine Information zu vermitteln. Dies soll klar und verständlich geschehen, am besten so, daß beispielsweise die Kameraführung einfach ihren Zweck erfüllt, ohne darüberhinaus aufzufallen. Um dies zu erreichen, muß sie in einem gewissen Sinne „richtig"oder „gut"sein, ein direkter künstlerischer Anspruch ist jedoch damit nicht verbunden, während wohl jeder Drehbuchautor, Kameramann oder Animateur diesen an sich selbst stellen wird.

1.2 Handhabung der Animationsbeispiele

Die Druckfassung dieser Arbeit enthält mehrere Animationsbeispiele in Form von Daumenkinos auf den freien Rückseiten der einseitig mit Text bedruckten Blätter. Diese Dokumentationsart entstand aus der Notwendigkeit, die grafischen Ergebnisse des Systems zu dokumentieren, gleichzeitig jedoch den Rahmen eines gedruckten und gebundenen Buches nicht zu verlassen.

Um die Daumenkinos ablaufen zu lassen, ist das Buch mit der Vorderseite nach unten und der Bindung auf der rechten Seite auf den Tisch zu legen (siehe Bild 2). Mit der rechten Hand greift man die nicht gebundene Seite des Buches

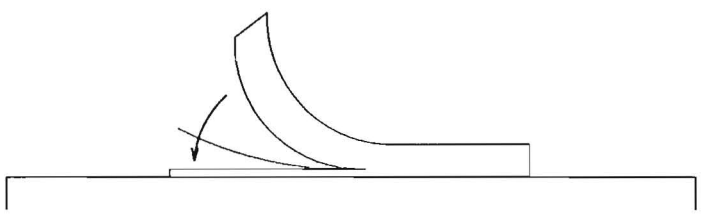
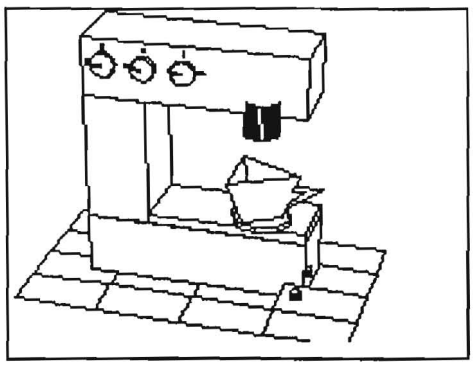
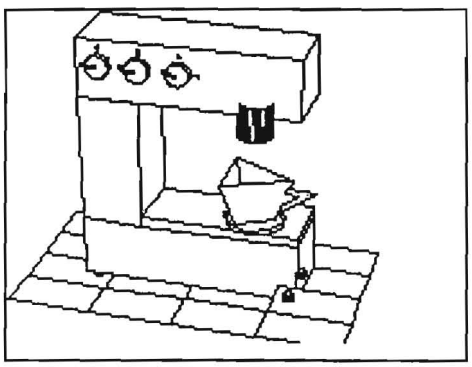
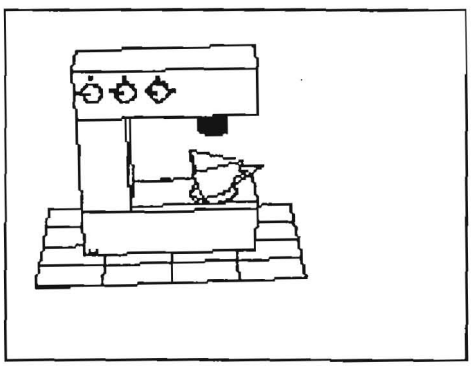


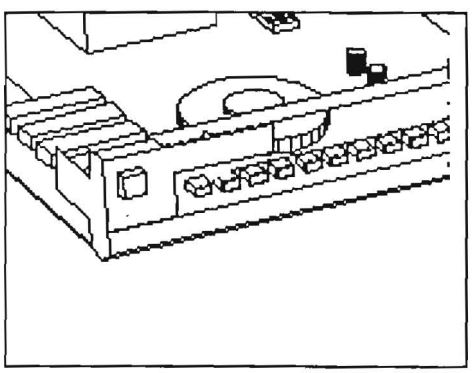
Abbildung 2: Abspielen der Daumenkinos

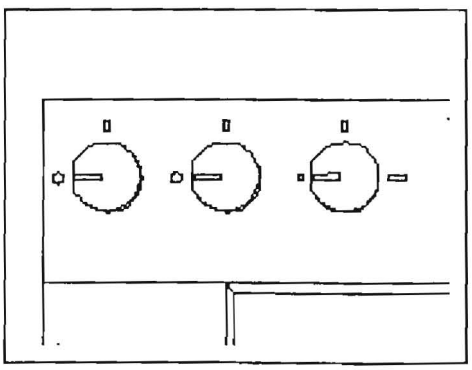
und biegt sie nach oben bis die Blattenden etwa senkrecht stehen. Der rechte Daumen läßt nun Seite um Seite in schneller Folge nach links umschlagen, so











daß Bild für Bild wieder auf dem Tisch zu liegen kommt. Währenddessen sollte sich das Auge auf eines der fünf ablaufenden Daumenkinos konzentrieren. Die Animationen sind von oben nach unten durchnumeriert und zeigen:

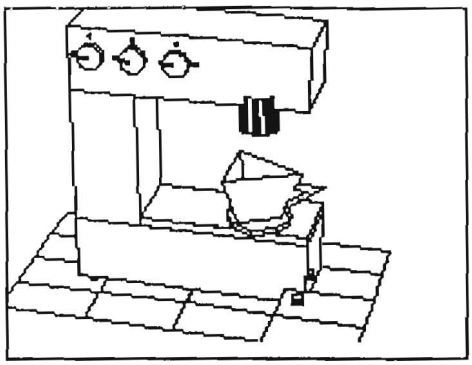
1. Verfolgung einer Objektbewegung mit der Kamera durch Fixieren des Objektes im Bild
2. Zeigen einer Objektbewegung unter Fixieren des Kamerazielpunktes im Raum
3. kombiniertes Verfahren zum Zeigen von Objektbewegungen, wie es in BETTYs Regelwerk codiert ist
4. Lokalisation zweier Objekte unter Berücksichtigung des visuellen Kontextes
5. Zeigen mehrerer Bewegungen in einer Animation.

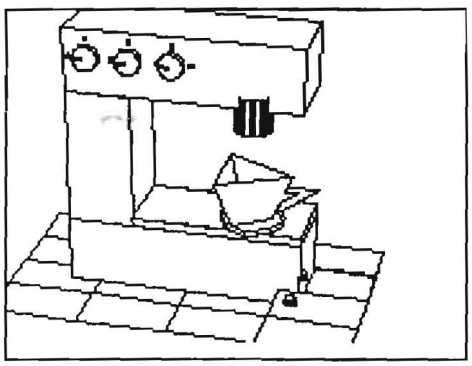
Auf die Besonderheiten der jeweiligen Inhalte oder angewandten Verfahren wird an gegebener Stelle im Text näher eingegangen.

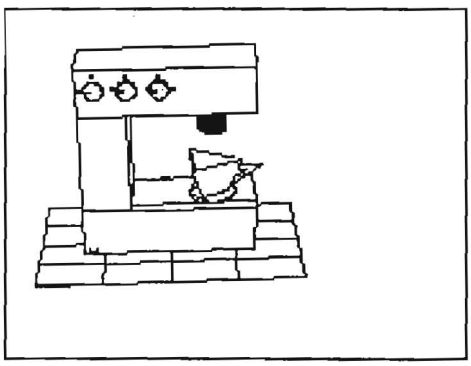
1.3 Terminologie

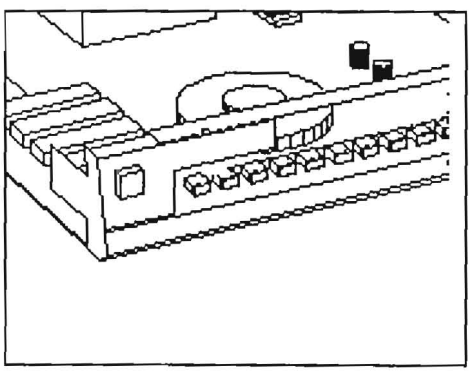
Um Mißverständnisse der von mir verwandten Terminologie von vorneherein auszuschließen, möchte ich hier einige für das Verständnis der Arbeit wichtige Begriffe kurz erklären.

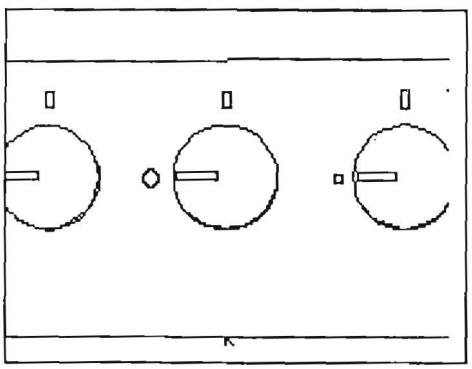
- *Präsentationsmedium* ist der physikalische Träger einer Präsentation, z.B. Bildschirm oder bedrucktes Papier.
- *Präsentationsmodus* ist das Kodierungsmittel der Information innerhalb des Mediums, z.B. Text oder Grafik auf bedrucktem Papier
- *Brennweite* bezeichnet einen (evtl. veränderlichen) Parameter eines Objektives. Bei einer Kamera wächst mit der Brennweite auch der Abbildungsmaßstab, und zwar direkt proportional.
- *Zoom* ist das Verändern der Objektivbrennweite der Kamera unabhängig vom Kamerastandort, was in einer Veränderung des Abbildungsmaßstabes resultiert.











- *Bildwinkel* eines Objektivs (horizontaler Bildwinkel, engl. view-angle) bezeichnet den Winkel im Raum, der von der Kamera aus gesehen zwischen rechtem und linkem Bildrand liegt. Der Bildwinkel verhält sich umgekehrt proportional zur Brennweite, also auch zum Abbildungsmaßstab eines Objektivs.

- *Normalbrennweite* ist die Brennweite, die der (je nach Filmformat unterschiedlichen) Diagonale des vom Objektiv entworfenen Bildes entspricht. Der daraus entstehende Bildwinkel von 45 Grad entspricht dem Blickwinkel beim Betrachten des Bildes, wenn man von einem Betrachtungsabstand ausgeht, der wiederum der Bilddiagonale des gezeigten Bildes entspricht. Unter dieser Grundannahme liegt die Normalbrennweite für Kleibildfilm etwa bei 50mm, für 6x6cm bei 80mm.

- *Weitwinkelobjektiv* ist ein Objektiv mit deutlich größerem Bildwinkel als dem der Normalbrennweite.

- *Teleobjektiv* ist ein Objektiv mit deutlich kleinerem Bildwinkel als dem der Normalbrennweite und infolgedessen bei gleicher Entfernung größerem Abbildungsmaßstab.

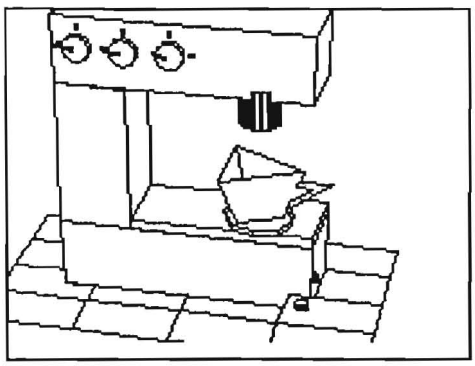
- *Zielpunkt* der Kamera ist der Punkt im Raum, der bei der mathematischen Projektion nach der Zentralperspektive in die Mitte des Bildes projiziert wird (verständlicher: der Punkt, auf den die Kamera gerichtet ist).

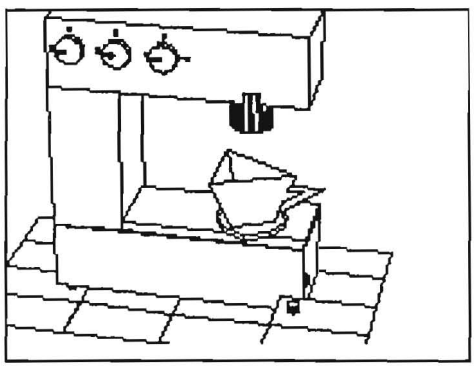
- *Move* oder Kamerafahrt ist die Veränderung der Kameraposition unabhängig von anderen Einstellungen.

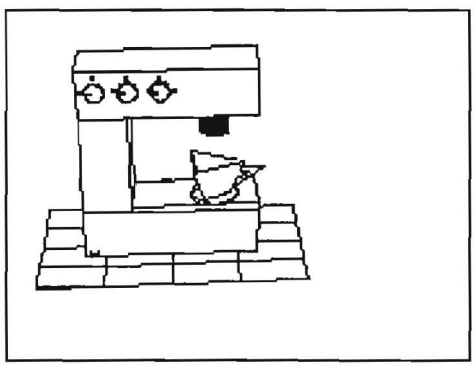
- *Pan* oder Kameraschwenk ist die Veränderung der optischen Achse der Kamera unabhängig von anderen Einstellungen, z.B. ein Drehen um die Vertikale.

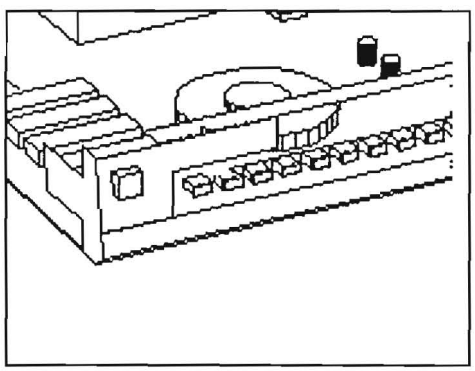
- *Cut* oder Schnitt bezeichnet die Grenze zwischen zwei Einstellungen, die als plötzliche Veränderung der Szenerie und/oder irgendwelcher Kameraeinstellungen sichtbar wird.

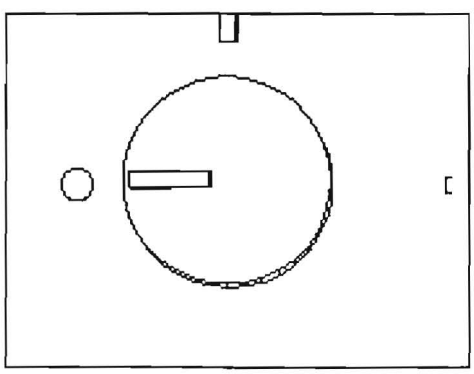
- *Einstellung* oder Shot ist ein zusammenhängendes Stück einer Szene, das ohne Unterbrechung gedreht wird.











- *Szene* ist eine Folge von Einstellungen innerhalb einer Sequenz, die einen sehr engen inhaltlichen Zusammenhang aufweisen und meist am gleichen Ort spielen.

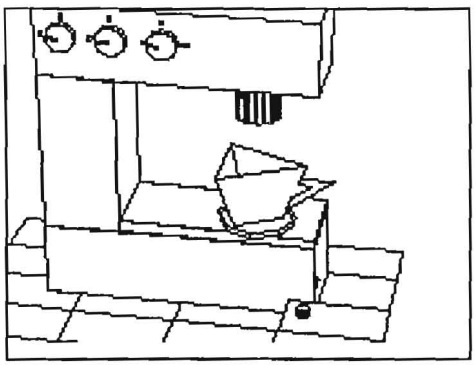
- *Sequenz* ist eine Folge von Szenen innerhalb eines Films, die einen inhaltlichen Zusammenhang bilden.

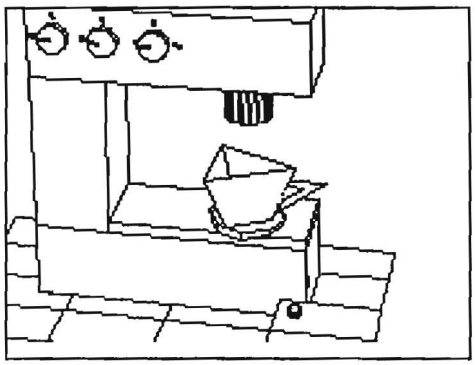
- *Weltszene*, wie in TOPAS verwandt, ist im Unterschied zur Filmszene eine Situation der Modellwelt, der Szenerie, und hat nichts mit einem zeitlichen Verlauf zu tun.

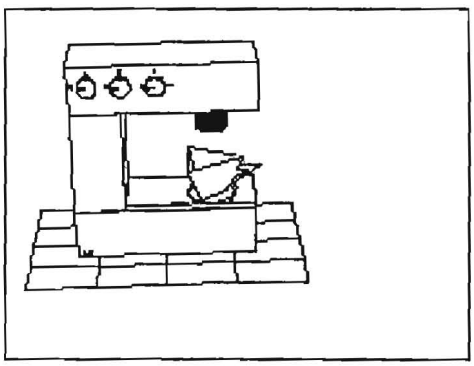
- *Viewport* bezeichnet das (evtl. nachbearbeitete) Bild einer Kamera, das in einen geeigneten Bildschirmbereich, z.B. ein Fenster in einer fensterorientierten Oberfläche projiziert wird. Eigenschaften des Viewports sind beispielsweise seine Position auf dem Bildschirm und die Einstellungen der zugehörigen Kamera.

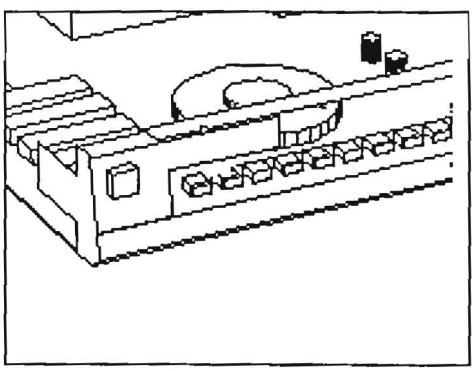
- *Zeitlupe* ist eine Darstellungsform, die Vorgänge langsamer wiedergibt, als sie in Wirklichkeit ablaufen und wird benutzt, um schnelle Vorgänge (z.B. ein Tor beim Fußball) klar erfaßbar zu machen.

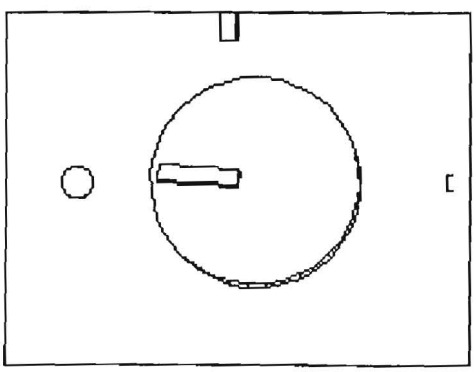
- *Zeitraffer* ist eine Darstellungsform, die Vorgänge schneller wiedergibt, als sie in Wirklichkeit ablaufen und wird benutzt, um sehr langsame Vorgänge klar erfaßbar zu machen (z.B. das Aufblühen einer Blüte).











2 Animation als Informationsträger

2.1 Was sind Animationen?

Der Begriff (Computer-) Animation (anima = lat. Seele, Animation = Beseelung, Belebung), umfaßt vom allgemeinsten Standpunkt aus jegliche Bewegung oder Veränderung beliebiger Objekte auf einem Computerbildschirm oder einem andersartigen geeigneten Ausgabemedium, seien dies Modelle von Weltobjekten in einem Viewport, der Viewport selbst auf einem Bildschirm, zu einem Rechner und seiner Benutzeroberfläche gehörende Objekte wie Fenster oder Icons, der Mauszeiger oder sogar einzelne Buchstaben oder Pixel. Von einem so weit gefaßten Verständnis ausgehend ließe sich die gesamte Ausgabe eines beliebigen Programmes als Animationsaufgabe formulieren. Da dies nun sicherlich nicht sinnvoll ist, möchte ich den Animationsbegriff, wie ich ihn für meine Arbeit verwenden werde, zunächst wie folgt einschränken:

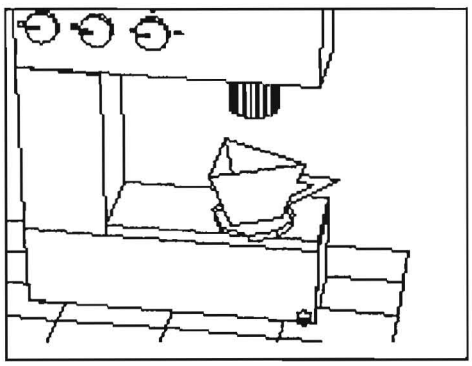
> 3D-Animationen im Rahmen dieser Arbeit sind Darstellungen geometrischer Modelle von Weltobjekten sowie deren Bewegungen in der Zeit auf einem Viewport (z.B. einem Fenster) am Bildschirm des Rechners in Form schnell aufeinanderfolgender Einzelbilder.

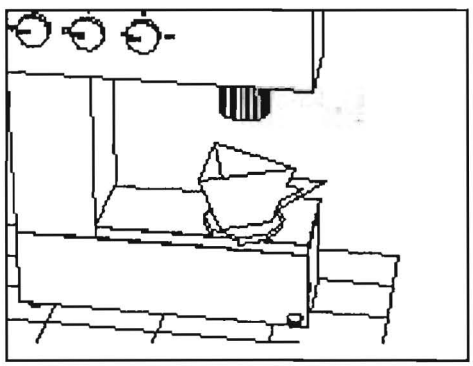
Mit zunehmender Geschwindigkeit und Speicherkapazität der zur Verfügung stehenden Rechner werden Animationen immer öfter verwandt und ihre Erzeugung wesentlich erleichtert. In der bildenden Kunst macht die Computeranimation im Bereich Neue Medien bereits eine eigene Sparte aus und eine wachsende Anzahl Künstler bedient sich dieses neuen Werkzeugs als künstlerischem Ausdruck.

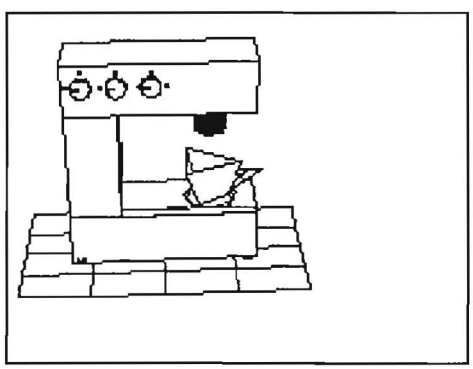
Ein anderes Anwendungsgebiet sind Hypermedia-Informationssysteme wie Hypercard oder XMosaic, in denen Animationen in Verbindung mit Text, Bild und Ton verwandt werden, um einem Benutzer Informationen zu vermitteln. Die dort benutzten Animationssequenzen sind jedoch bislang ausnahmslos vorgefertigte und abgespeicherte Bildfolgen, deren Einzelbilder zwar vom Computer berechnet sind, deren Ablauf jedoch von Menschen gestaltet wurde.

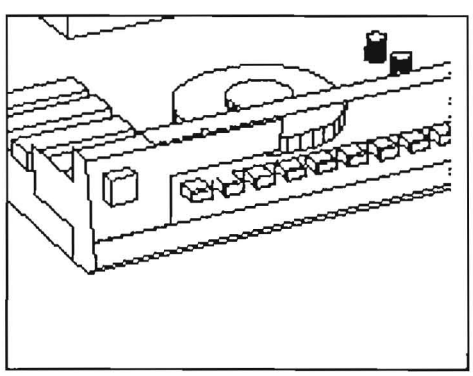
2.2 Was können Animationen?

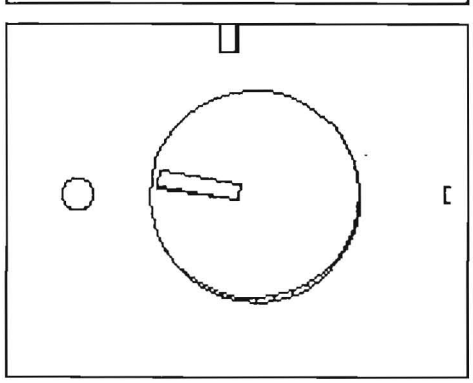
Die Darstellung eines Inhaltes in Form einer Animation ist zunächst einmal überall dort möglich, wo der Inhalt eine geometrische Veränderung eines darstellbaren Objekts bedeutet und in der Zeit abläuft. Das wäre z.B. die Bewegung von Teilen,











wobei das Gewicht bei einer Darstellung als Animation meist auf der Bewegung selbst und nicht auf Anfangs- und Endzustand liegt, außerdem das Darstellen dynamischer räumlicher Relationen, z.B. Trajektorien, oder das Darstellen zeitlicher Relationen von Vorgängen, sprich der Reihenfolge oder allgemeiner, der Verhältnisse zwischen Zeiten.

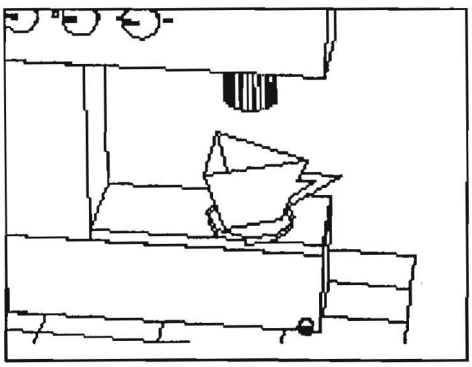
Ferner läßt sich mit dem Mittel der Animation auch statische Information dynamisch verdeutlichen. Soll beispielsweise ein Detail eines Objektes lokalisiert werden, z.B. ein Schalter an einem technischen Gerät, so kann man dies durch eine Filmsequenz erreichen, die zunächst das Objekt insgesamt in einer Standardperspektive zeigt (um den visuellen Kontext zu schaffen), anschließend eine Kamerafahrt sowie ein Zoom auf das Detail zu und ein Verharren in der Zielposition, in der das Detail zu erkennen ist. Diese Endeinstellung wäre als Standbild ohne die vorangehende Kamerafahrt zur Vermittlung der Gesamtsituation nicht ausreichend, da bei genügend sichtbarem Umfeld das Detail zu klein dargestellt würde, bei ausreichender Größe des Details aber der Bezug zum Ganzen nicht mehr vorhanden wäre. In statischen Grafiken wurde dieses Problem z.B. in Form einer Ausschnittsvergrößerung, eines Insets gelöst.

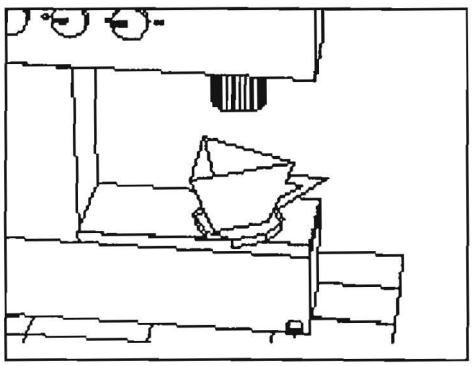
So gibt es also neben den Inhalten, die sich eigentlich nur in einem dynamischen Medium bzw. Modus adäquat darstellen lassen, auch einen Bereich, in dem sich die Verwendbarkeit statischer und dynamischer Medien und Modi überschneiden. Hier muß eine Auswahl nach anderen Kriterien, z.B. Kontextbedingungen und ästhetische Kriterien in der Gestaltung der gesamten Präsentation erfolgen.

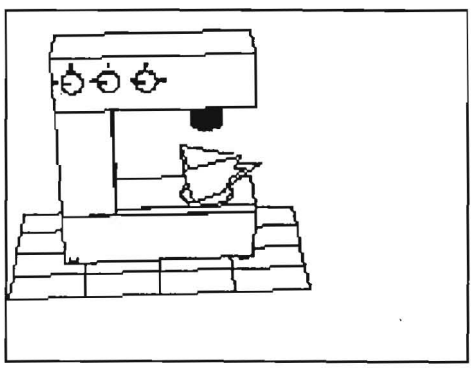
2.2.1 Die drei räumlichen Dimensionen

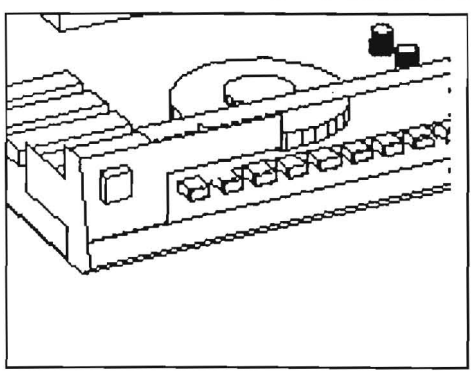
Unser Leben und sämtliche Vorgänge in unserer Umgebung spielen sich in einem vierdimensionalen Raum-Zeit-Kontinuum ab. Das Medium Bildschirm erlaubt eine Darstellung von Vorgängen jedoch nur in zwei räumlichen Dimensionen, nämlich der Horizontalen und der Vertikalen der Bildfläche, sowie in der Dimension Zeit. Sollen dreidimensionale Objekte und Szenen dargestellt werden, so muß die dritte Raumdimension mittels eines Tricks dargestellt, das heißt auf die anderen Dimensionen projiziert werden. Hierzu gibt es prinzipiell mehrere Möglichkeiten, wovon in unserem Kulturkreis die sogenannte Zentralprojektion die gebräuchlichste ist (siehe auch Bild 8).

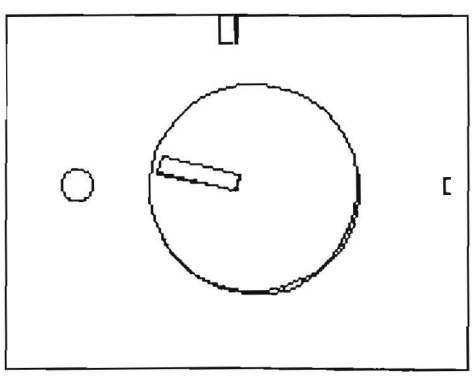
Diese rechnerische Projektion der drei Raumdimensionen auf zwei Bildschirmdimensionen liefert das gleiche Abbild eines modellierten Körpers, wie ein fotografisches Objektiv es von dem entsprechenden Körper im Raum entwerfen würde (siehe [Ma88]).



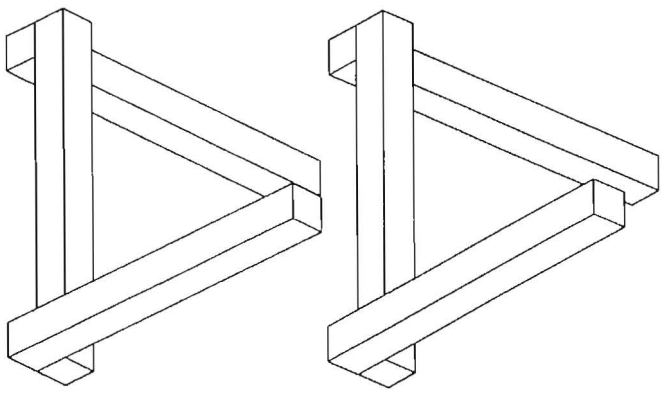








Unser Gehirn hat gelernt, aus solchen Abbildungen wieder einen dreidimensionalen Körper zu rekonstruieren, was keineswegs angeboren oder eine der Methode inhärente Eigenschaft ist. Kulturen, die mit unserer technisierten Welt bisher nicht in Berührung kamen, können nämlich teilweise weder mit Fotos noch anderen zweidimensionalen Darstellungen dreidimensionaler Dinge etwas anfangen, aber auch wir scheitern zuweilen, wie die beiden folgenden Bilder zeigen.

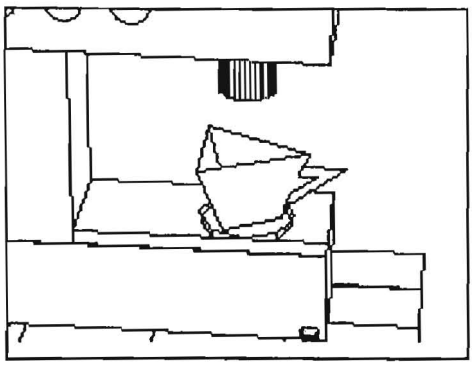


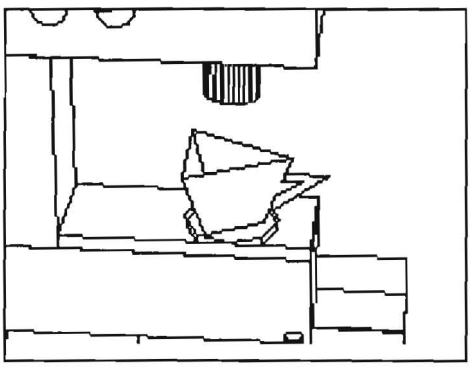
Linkes Auge Rechtes Auge

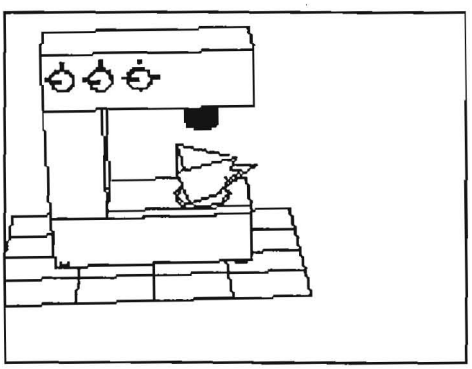
Abbildung 3: Verschiedene Projektionen durch versch. Aufnahmepositionen

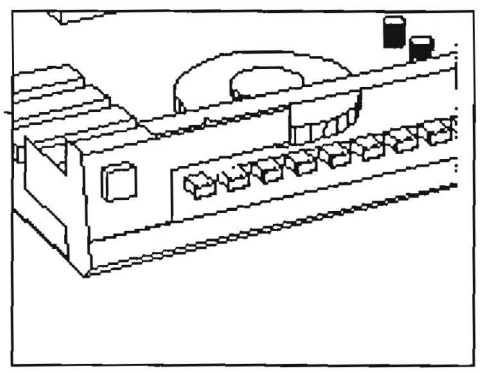
Betrachten wir das linke Bild einzeln, so können wir uns auf die (mathematisch korrekte) Projektion keinen Reim machen. Betrachten wir nun das rechte Bild, so wird (durch eine leichte Veränderung der Kameraposition) die dreidimensionale Form rekonstruierbar. Gehen wir schließlich mit dem Gesicht bis auf etwa 10-15cm an das Papier heran und lassen beide Figuren übereinanderwandern (indem wir uns vorstellen, ganz weit weg durch das Papier hindurch zu schauen), so entsteht sogar ein richtig dreidimensionales Bild (die mittlere der drei dann erscheinenden Figuren).

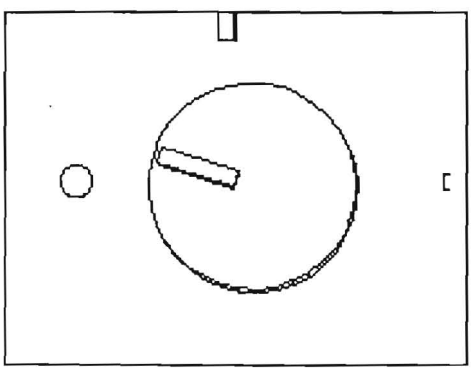
Um in der wirklichen Welt dreidimensional sehen zu können, braucht das Gehirn die Summe der Informationen zweier Augen, die in hinreichendem Abstand zueinander angebracht sind. Durch die verschiedenen Positionen dieser beiden „natürlichen Kameras"werfen beide ein etwas unterschiedliches Bild, und aus diesem Unterschied der Bilder rekonstruiert das Gehirn die Tiefeninformation, so











daß wieder ein dreidimensionales „Bild"entsteht.

Fällt ein Auge aus, so kann eine ähnliche Wirkung durch Hin- und Herbewegen des Kopfes erreicht werden, was ebenfalls einer Verlagerung des Kamerastandpunktes entspricht. Dieses Verfahren wurde übrigens versuchsweise benutzt, um dreidimensionale Bilder im Fernsehen ohne Verwendung einer Spezialbrille zu übertragen.

Grundsätzlich bietet jedes dynamische Medium wie Film, Fernsehen oder Animation schon vom Prinzip her den entscheidenden Vorteil einer veränderlichen Kameraposition innerhalb einer Darstellung gegenüber beispielsweise der Fotografie oder technischen Grafik. Fährt eine Filmkamera teilweise um ein Objekt herum, so ist es für den Betrachter in natürlicher Weise möglich, aus den vielen Einzelbildern ein dreidimensionales Abbild des gezeigten Gegenstandes zu rekonstruieren.

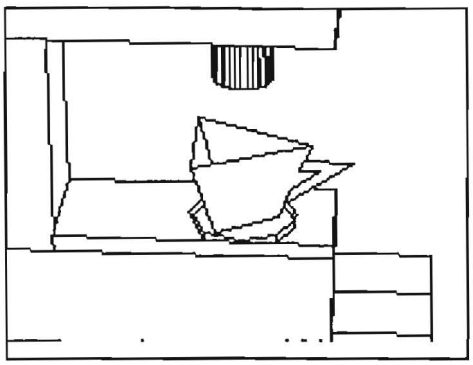
In Wirklichkeit bedeutet dies, daß die dritte Raumdimension zum Teil in die Dimension Zeit projiziert wird, das heißt, daß eigentlich eine Projektion des vierdimensionalen Raum-Zeit-Kontinuums in ein dreidimensionales Fläche-Zeit-Kontinuum stattfindet.

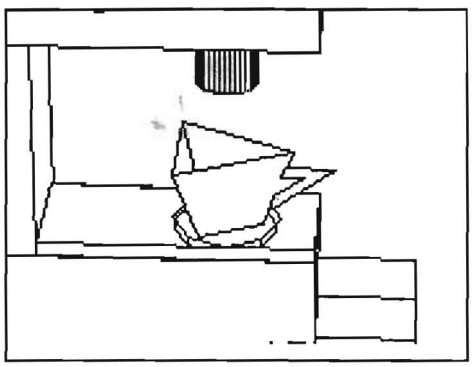
2.2.2 Die Dimension Zeit

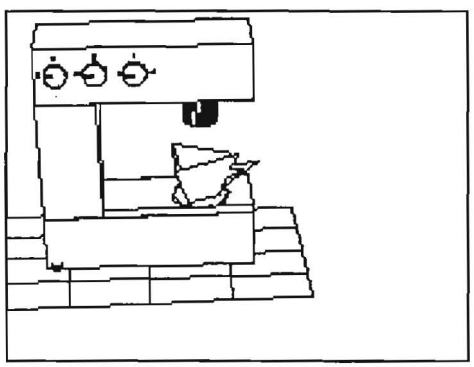
Auch die Dimension Zeit wird in statischen Grafiken durch Hilfsmittel ausgedrückt, wie explizite Zeitangaben, Numerierungen oder Bildfolgen. Die vierte Dimension muß also ebenfalls in die beiden Flächendimensionen des Bildschirms projiziert werden, und zwar mittels mehr oder weniger starker Umschreibungen.

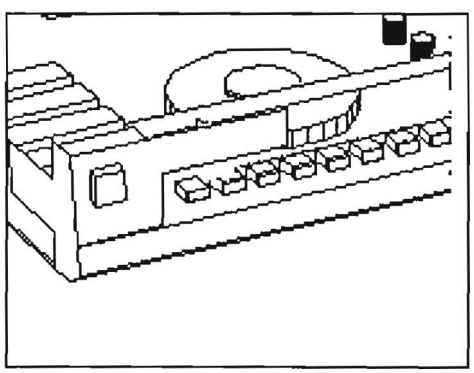
Im dynamischen Modus Animation ist die Dimension Zeit genauso Bestandteil der Präsentation wie die beiden Flächendimensionen und braucht daher prinzipiell nicht auf irgendeine andere Dimension abgebildet zu werden. Dabei ist allerdings unsere Sehgewohnheit in gewisser Weise zu beachten. Man kann z.B. keinen mehrstündigen Vorgang in einer Animation von wenigen Sekunden naturgetreu zeigen. Die Präsentationszeit bestimmt die Dauer der Darstellung, genau wie der verfügbare Platz ihre Größe bestimmt.

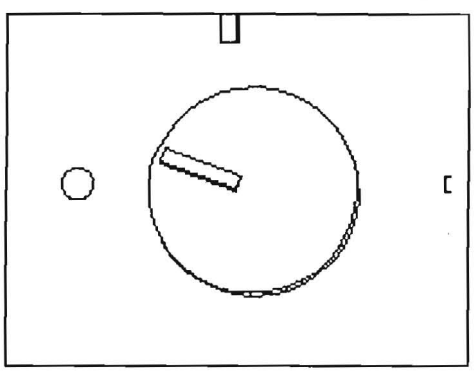
Diese notwendige Veränderung des „Zeitmaßstabes"entspricht aber genau der Veränderung des räumlichen Maßstabes, da wir ja auch beliebig große Objekte nur verkleinert auf dem Bildschirm darstellen können. Allerdings ist unsere Wahrnehmung mit einer Veränderung der Größe vertrauter als mit einer Veränderung der Zeit. Wir wissen schließlich, wie groß ein prototypisches Auto ist und stören uns nicht daran, daß es auf dem Papier nur noch $5cm$ mißt. Legt es hingegen in einer Animation innerhalb weniger Sekunden eine riesige Strecke zurück, so











fragen wir uns wohl, ob es sich um ein besonders schnelles Auto handelt, obwohl wir sehr wohl wissen, wie schnell ein prototypisches Auto fährt. Zeitlupe und Zeitraffer sind Mittel, die wir aufgrund unserer Sehgewohnheiten eher mit einer Semantik belegen, als sie einfach wie eine Veränderung des räumlichen Maßstabes hinzunehmen.

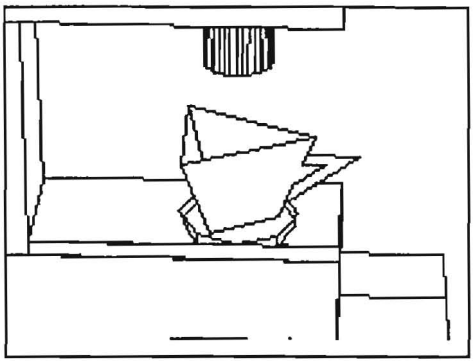
Trotzdem bleibt der Animation eine viel direktere Darstellung der Zeit vorbehalten, da unabhängig von einer zeitlichen Maßstabsveränderung die Verhältnisse zwischen den Zeitintervallen erhalten bleiben. Die Reihenfolge von Aktionen sowie zeitliche Überschneidungen oder Pausen zwischen Aktionen sind Elemente, die in einer Animation ohne Umschreibung direkt visualisiert werden können.

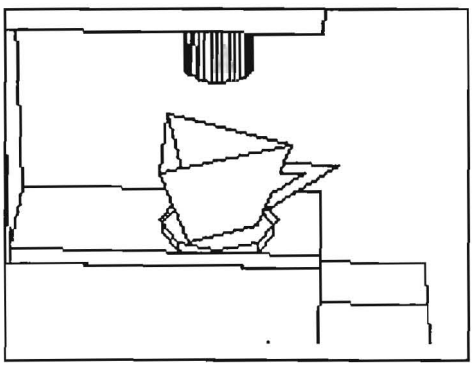
2.2.3 Die Präsentation einer Animation in Raum und Zeit

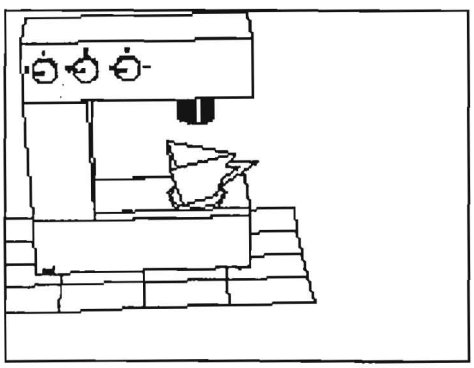
Bei der Ausgabe dynamischer Dokumente, die Animationen enthalten, stellt sich ein grundlegendes Problem, nämlich die Einbeziehung des Faktors Zeit in die Gestaltung des Dokumentes. Herkömmliche multimodale Dokumente ohne Animation sind (zumindest in der Ausgabe) statisch bzw. inkrementell (sofern wir Hypermedia-Präsentationen ausschließen). Sie lassen sich gleichermaßen auf dem Bildschirm ausgeben wie auch auf Papier gedruckt, nutzen also die 3. Dimension Zeit, die der Bildschirm im Vergleich zu den 2 Dimensionen des Papiers bietet, nicht aus. Eine Animation läuft aber gerade in dieser 3. Dimension ab und verlangt also, sie beim gesamten Gestaltungsprozeß mit zu berücksichtigen. Wie soll das System beispielsweise entscheiden, wann der Leser mit dem Erfassen eines vorangehenden Textes fertig ist und die Animation ablaufen kann? Wie sollen Bezüge zwischen Animation und begleitendem Text gehandhabt werden?

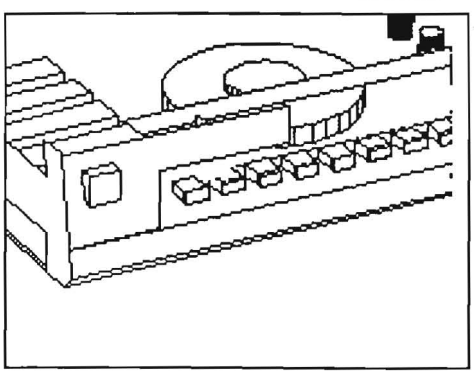
Während sich Text und (statische) Grafik durch die bloße Abfolge und Anordnung im Layout eines Dokumentes räumlich koordinieren lassen und somit Querbezüge ermöglichen, ist die zeitliche Koordination einer Animation mit nicht zeitgebundenen Modi wesentlich schwieriger zu bewerkstelligen. Während eine statische Grafik dem Betrachter beliebig Zeit läßt, mit dem Blick zum Text zu wandern, dort etwas zu lesen und die in der Zwischenzeit ja nicht veränderte Grafik daraufhin von neuem zu betrachten, läuft die Animation auch weiter, wenn der Betrachter gerade nicht hinschaut.

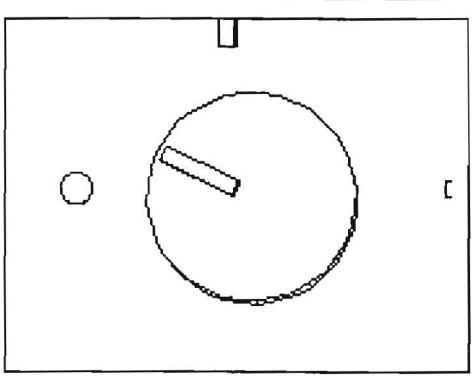
Soll ein als Animation dargestellter Vorgang zusätzlich durch natürliche Sprache beschrieben oder ergänzt werden oder umgekehrt ein gegebener Text durch Animation illustriert werden, so gibt es u.a. folgende Synchronisationsmöglichkeiten: Man gibt es dem Benutzer in die Hand, den Ablauf der Animation zu beeinflussen, d.h. sie nach Belieben vor- und zurücklaufen zu lassen. So wird es











ermöglicht, jeweils auch das sinngemäß zugehörige Stück Text (z.B. als Untertitel) darzustellen und so beides zu koordinieren. Das Auge und die Aufmerksamkeit des Betrachtes muß dann ständig zwischen Animation und Text wechseln, außerdem muß eine Hand an der Maus oder Tastatur bleiben.

Die nächste Möglichkeit ist, den Betrachter zuerst den gesamten Text lesen zu lassen und ihm die Gelegenheit zu geben, dann die Animation zu starten (wie es in den bisherigen Hypermedia-Systemen der Fall ist). Schließlich könnte eine kurze Bildsequenz auch immer wieder ablaufen, so daß der Benutzer etwas Text liest und dann an einer zufälligen Stelle in die laufende Animation einsteigt.

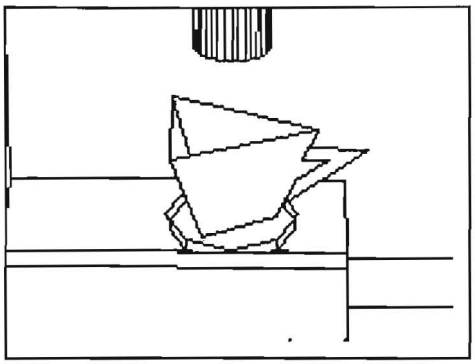
Bildfolgen und gesprochene Sprache

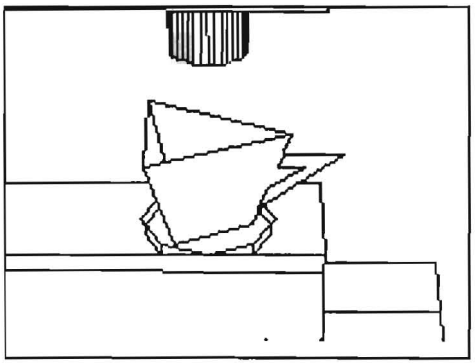
Die beste Koordination von Animation und Sprache ist jedoch aus einem offensichtlichen Grund die Ergänzung durch gesprochene Sprache, deren Ausgabe zeitlich mit dem Ablauf der Animation synchronisiert ist. Der wichtige Unterschied zu anderen Koordinationsmöglichkeiten ist der, daß in der gleichen Zeit zwei verschiedene Sinne – das Gehör und das Auge – angesprochen werden, die unabhängig voneinander gleichzeitig aufnahmefähig sind. Dieser Effekt wird umgekehrt auch in einem von [Pe91, Pe92] beschriebenen Projekt untersucht, um durch die animierte Darstellung eines Gesichts mit entsprechenden Lippenbewegungen das Verstehen gesprochener Sprache zu unterstützen. Die gleichzeitige Nutzung beider Modi verstärkt auch hier die Verständlichkeit und Ausdruckskraft der Präsentation.

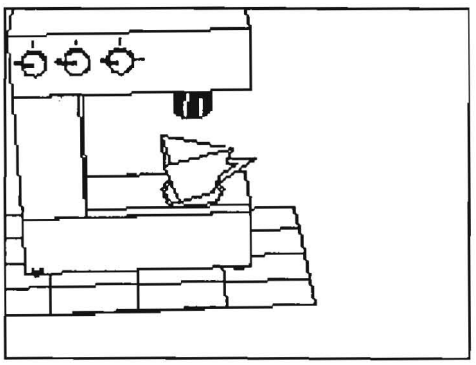
In jedem Falle aber erfordert die Integration einer Computeranimation in ein am Bildschirm dargestelltes Dokument eine grundlegend neue Sichtweise des Layouts, da die Darstellung einer Information als Animation dem Rest des Dokumentes die oben geschilderten zeitlichen Constraints auferlegt. Diese Problematik wird näher in [Gr93] besprochen. Grundvoraussetzung für eine sinnvolle Koordination verschiedener Modi im Layout ist die Einbeziehung der Dimension Zeit in die gesamte Gestaltung der Präsentation.

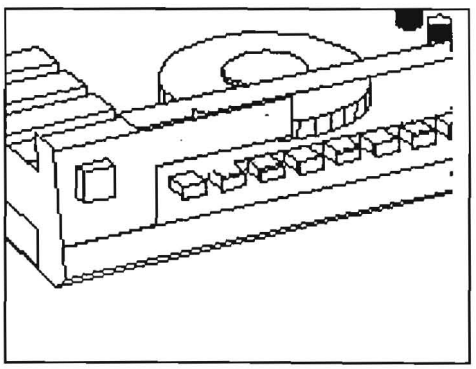
2.3 Abspeichern oder generieren?

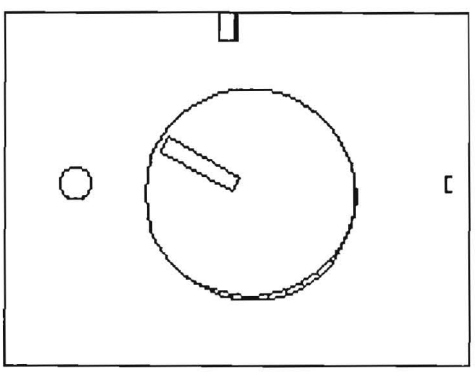
In die Berechnung realitätsnaher 3D-Grafiken mit Licht und Farbe sowie in die flüssige Darstellung der so erzeugten Einzelbilder als Animationen wurde bereits ein beträchtlicher Forschungsaufwand investiert (siehe hierzu [BBZ90, Fo90]) und diese Probleme sind weitgehend bereits gelöst. Anders sieht es jedoch bei der Gestaltung der Animationsdrehbücher aus. Die gekonnte Kameraführung sowie











das Setzen der Beleuchtung erfordert mit den bisher bestehenden Grafik- und Animationssystemen ein hohes Maß an Erfahrung und Detailarbeit.

Eine Arbeit, deren Ziel es ebenfalls ist, Animationen auf einem höheren Abstraktionsniveau beschreiben zu können, wird in [Br94] vorgestellt und bezieht sich vor allem auf die Beschreibung und Verknüpfung von Bewegungen.

Während in den letzten Jahrzehnten bei der automatischen Generierung natürlicher Sprache grundlegende Fortschritte erzielt und viele neue Methoden entwickelt wurden, wurde die inhaltliche Gestaltung von Animationen bis vor wenigen Jahren im Hinblick auf ihre Automatisierbarkeit überhaupt nicht untersucht. Die Verbesserung der graphischen Ausgabe im Modus Animation entspricht dabei lediglich einer Verschönerung des Druckbildes im Modus Text, um es drastisch auszudrücken. An der abstrakteren Gestaltung von Animationssequenzen, der Planung von Kamera- und Lichtführung und der Erzeugung regelrechter Drehbücher jedoch wird erst in den letzten Jahren verstärkt gearbeitet.

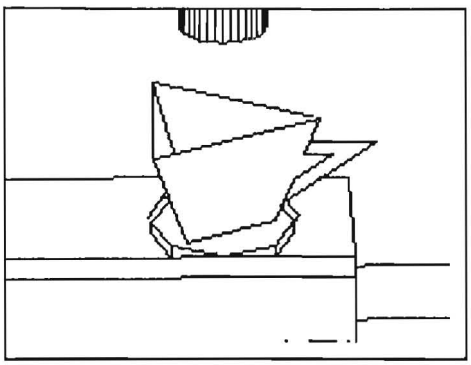
Bei der derzeit entwickelten Generation von Informations- und Präsentationssystemen ([Ka93, Gr92, Wa93]) ist nun die Präsentationsform der darzustellenden Information zum Designzeitpunkt des Systems oft gar nicht vorhersehbar. Die gleiche Information kann je nach Präsentationssituation zu völlig verschiedenen Präsentationen führen, die dann an die zu verwendenden Animationen die verschiedensten Anforderungen stellen.

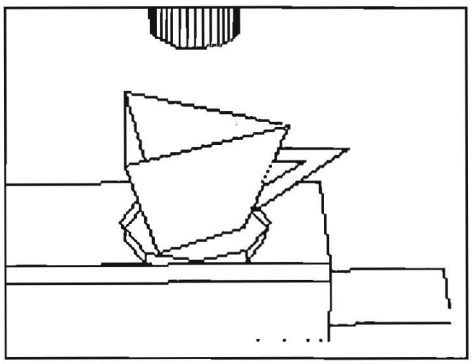
Im Modus Animation nur mit vorgefertigten, abgespeicherten Konserven zu arbeiten, bedeutet in etwa das gleiche, wie im Modus Text die Aneinanderreihung vorgefertigter Standardsätze. Diese Methode hat zwar bei eingeschränkten Ressourcen und einfachen Problemen ihre Berechtigung und findet in dieser Form auch allenthalben Anwendung, doch stößt sie bei komplexeren Inhalten sehr schnell an ihre Grenzen. Außerdem beschränkt sie sehr stark die Flexibilität und die Anpaßbarkeit der Präsentation an verschiedene Situationen.

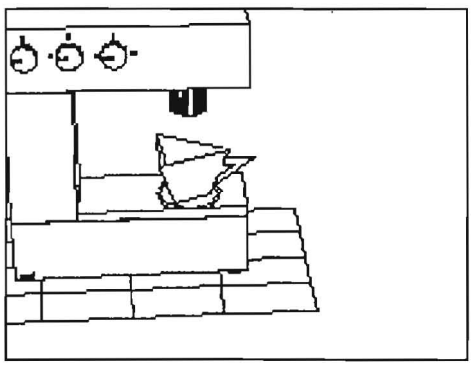
2.4 Eine Beschreibungshierarchie für Animationen

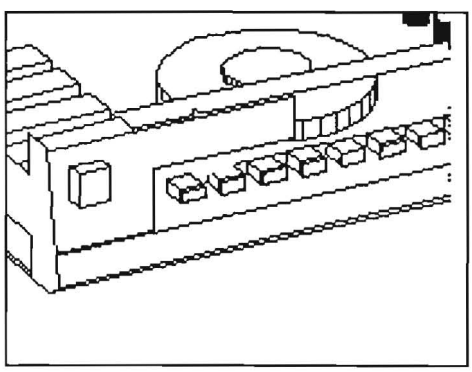
In Anlehnung an [Ze90] lassen sich auch Animationen zum Zweck der Informationspräsentation bzw. die sie beschreibenden Drehbücher oder Scripts auf verschiedenen Ebenen beschreiben. Oberste Stufe ist in Zeltzers Terminologie die Spezifikation der Aufgabe der Animation (task level). Man könnte zum Beispiel spezifizieren:

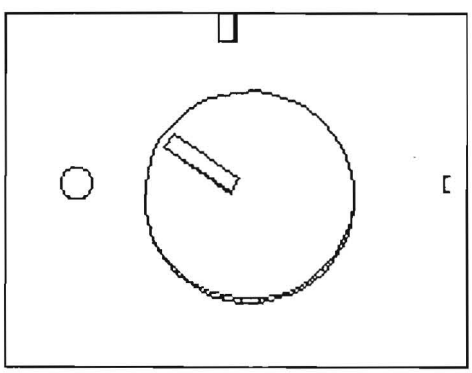
„Eine Animation, die die Bedienung des An/Aus-Schalters zeigt"











Die Erfüllung dieser Aufgabe wird durch filmtechnische Mittel erreicht (functional level), was zum Beispiel lauten könnte:

> *„Eine Animation, in der zunächst der An/Aus-Schalter ins Bild gebracht und anschließend seine Bewegung gezeigt wird"*

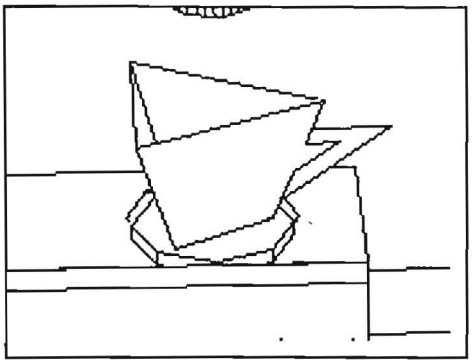
Diese filmtechnischen Mittel werden ihrerseits durch Folgen elementarer Regieanweisungen wie Schwenk oder Zoom beschrieben (procedural level). Beispiel:

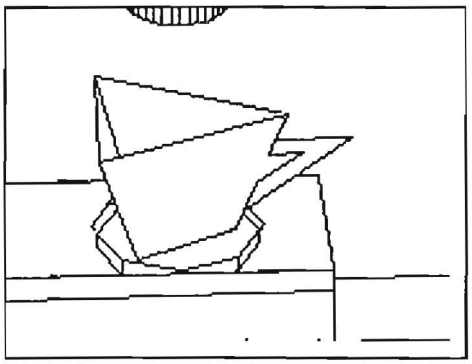
> *„Kamerafahrt an die rechte Seite des Gerätes, Zoom auf den An/Aus-Schalter, kurzes Standbild, Bewegung des Schalters gemäß der Aktion Einschalten, kurzes Standbild"*

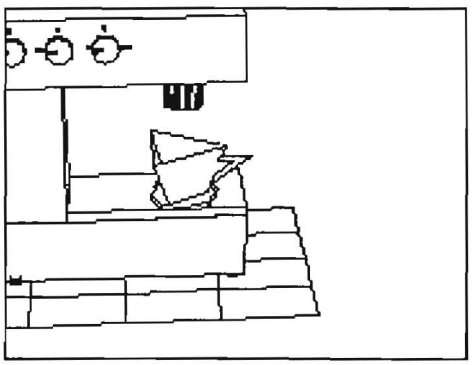
Auf unterster Ebene schließlich müssen alle elementaren Objekt- und Kamerabewegungen explizit angegeben werden (machine level). Das bedeutet eine Spezifikation aller elementaren Objektbewegungen entlang Achsen oder Trajektorien sowie aller Zustandsänderungen der Objekte oder der Kamera. Das zu den oben ausgeführten analoge Beispiel zeigt Bild 4. Die Objektbewegungen sind hier über den gesamten Ablauf der Animation exakt vorgegeben, und zwar in Form einer systemunabhängigen Beschreibungssprache, die in BETTY für die Beschreibung von Animationen verwandt wird (vgl. Kap. 4.4). Anhand des Rollbalkens am linken Rand ist zu ersehen, welcher Teil der Gesamtspezifikation sichtbar ist.

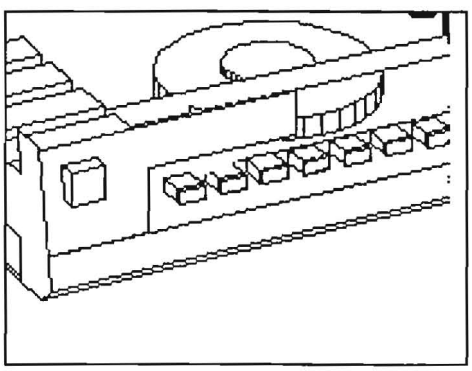
Ein solches Script ist hierarchisch aufgebaut und besteht aus mehreren Untersequenzen, die jeweils parallel oder aufeinanderfolgend verlaufen und ihrerseits wieder genauso in Untersequenzen zerfallen können. So läuft z.B. zu einer Sequenz, in der ein Objekt bewegt wird, mindestens eine weitere parallel, die die zugehörige Kameraführung beschreibt. An den Blättern dieses Untersequenz-Baumes befinden sich elementare Bewegungen (Translation, Rotation) der Kamera oder der Objekte zu bestimmten Zeiten entlang bestimmter Achsen oder Trajektorien.

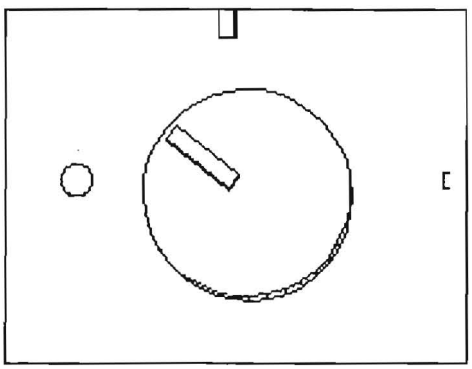
Wie wir später sehen werden, ist die Vorstellung des hierarchischen Aufbaus grundlegend für das Verständnis des in BETTY verwandten Planungsmechanismus, da dieser Aufbau bereits eine ebenso hierarchisch geschachtelte Datenstruktur zur Beschreibung der Animationsscripte nahelegt. Diese Datenstruktur wiederum läßt sich auf sehr elegante Weise mit dem später vorgestellten Planungsalgorithmus erzeugen.











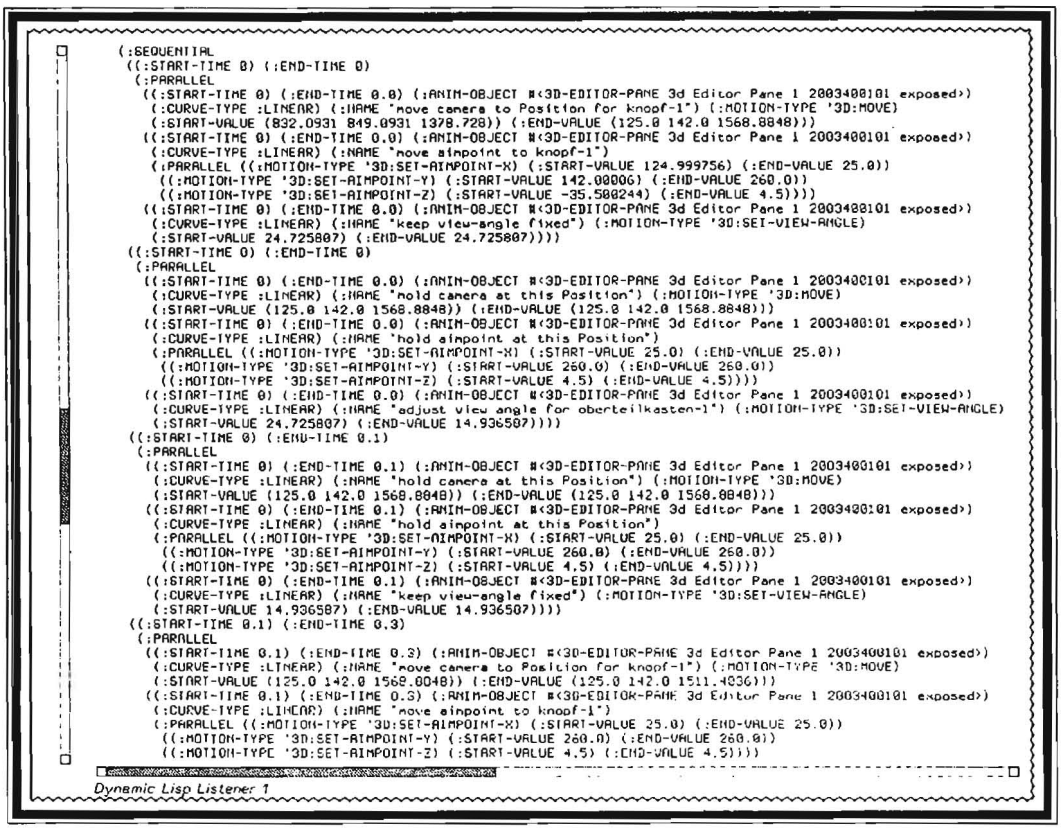


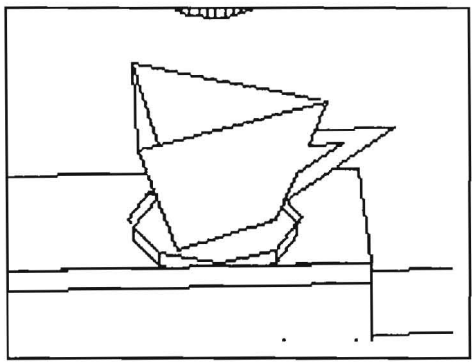
Abbildung 4: Vollst. Spezifikation aller Bewegungen zu jedem Zeitpunkt

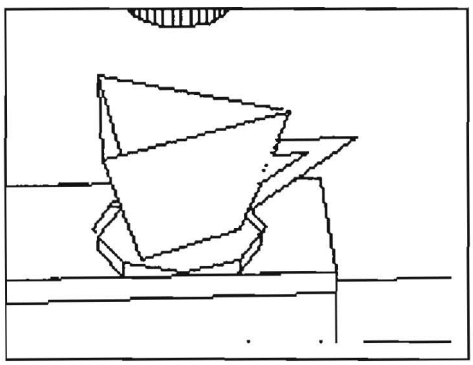
2.5 Animationen in WIP

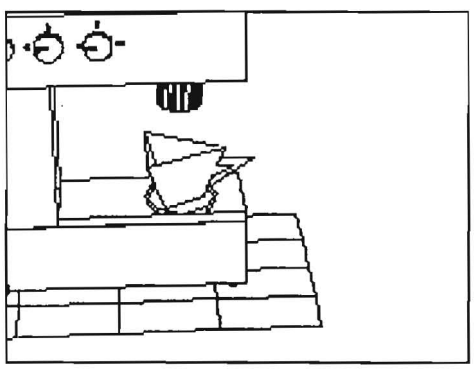
Das Projekt WIP (Wissensbasierte Informations-Präsentation, [Wa93]) am DFKI in Saarbrücken, in dem diese Arbeit entstand, sowie das Nachfolgeprojekt $\mathcal{PPP}$ ([PPP]) behandeln die planbasierte multimodale Präsentation von Information unter bestimmten Vorgaben (Zielgruppe, Umfang) auf verschiedenen Ausgabemedien (Bildschirm, Papier) in verschiedenen Ausgabemodi wie Text, (statische) Grafik und Animation. Konkret werden aus geometrischen Modellen technischer Geräte sowie Wissen über deren Aufbau und die mit ihnen ausführbaren Handlungen an verschiedene Situationen angepaßte Bedienungsanleitungen generiert.

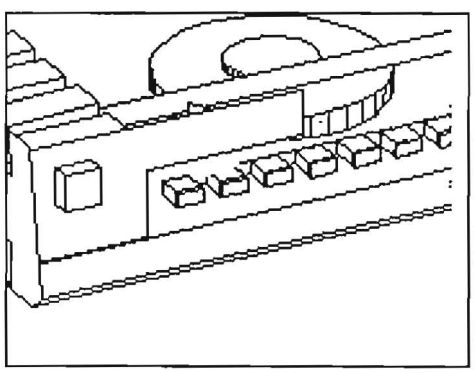
Der prinzipielle Aufbau des WIP -Systems ist in Bild 5 dargestellt.[1] BETTY

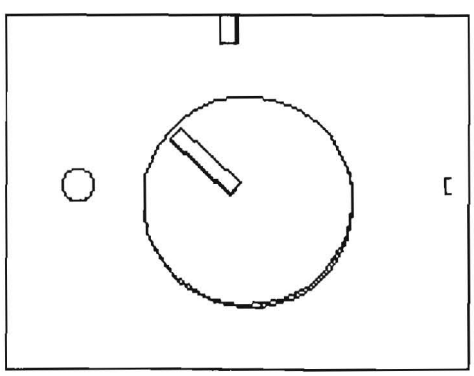
[1] Diese Integration BETTYs in das WIP System ist zwar derzeit nicht realisiert, jedoch wurde BETTY mit Hinblick auf eine solche Integration in WIP entwickelt.











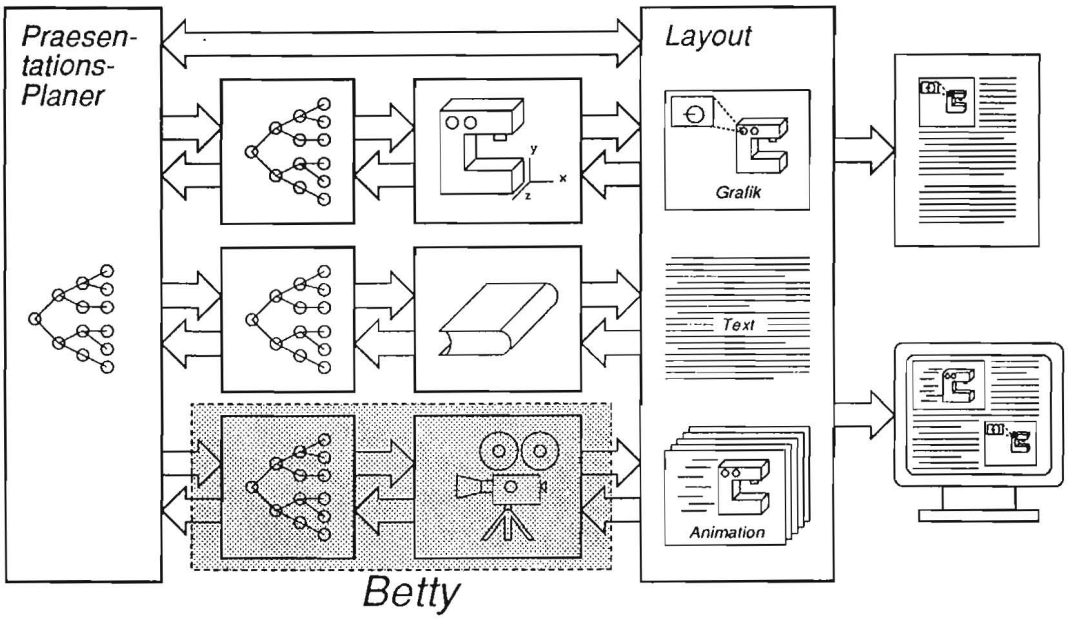


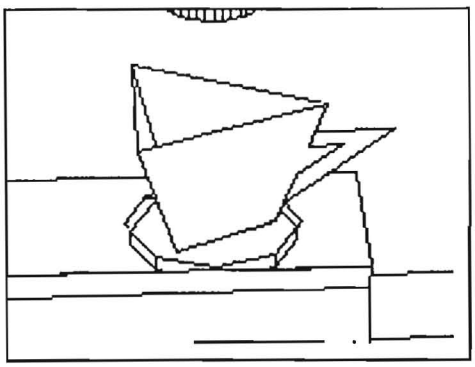
Abbildung 5: Aufbau des WIP -Systems

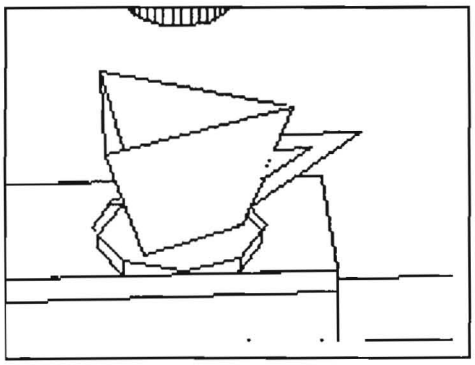
stellt hierbei die Planungs- und Realisierungskomponente für Präsentationen im Modus 3D-Animation dar. Ihr werden im Gesamtplanungsprozeß zu präsentierende Informationen zugewiesen, für die der Modus 3D-Animation als der geeigneteste angenommen wird. BETTY übernimmt dabei die komplette Gestaltung und Berechnung der Animation.

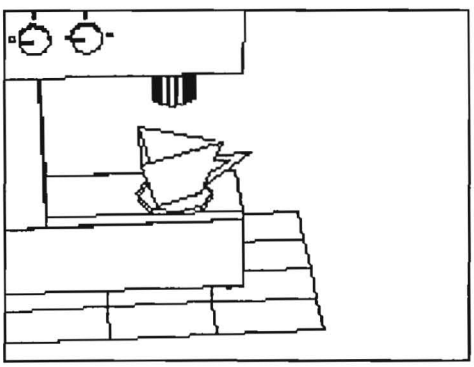
2.5.1 Spezifikation der Visualisierungsziele

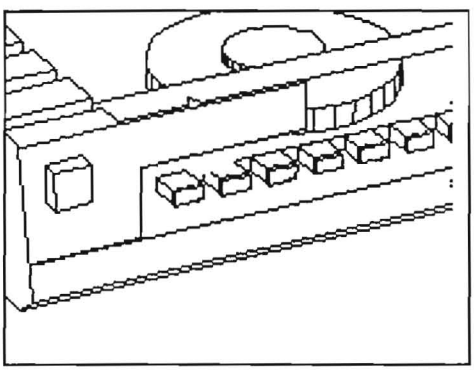
Die Präsentation von Informationen wird in WIP von einem Präsentationsplaner gestaltet (vgl. [An93]), der ausgehend von einem komplexen Präsentationsziel (z.B.: Der Benutzer soll die Bedienung eines bestimmten Schalters kennenlernen) eine hierarchische Struktur aufbaut. Bei den Blättern dieser Struktur handelt es sich um elementare Präsentationsaufgaben, die von den jeweiligen Generierungskomponenten in Präsentationsbestandteile umgesetzt werden.

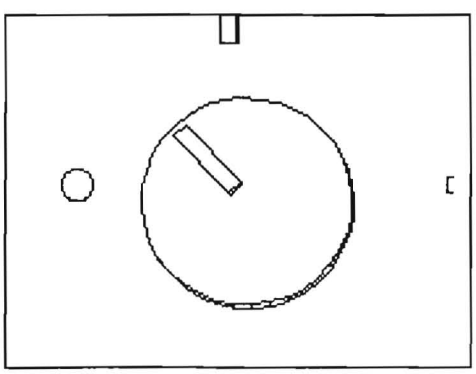
Darüberhinaus legt der Plan fest, durch welchen Modus die Präsentation dieser Information erfolgen soll. Die Position des Schalters könnte zum Beispiel in natürlicher Sprache beschrieben werden oder in einer Grafik oder einer Animation











gezeigt werden. Da es sich bei der Präsentation als Grafik oder Animation genaugenommen um eine Visualisierung der Information handelt, bezeichne ich diese speziellen Präsentationsaufgaben im folgenden auch als Visualisierungsziele.

Die Visualisierungsziele bilden die Eingabe an die Gestaltungs- und Realisierungskomponente zum Modus Animation (siehe wieder Bild 5). Sie stellen eine inhaltliche Beschreibung der Animation auf oberster Abstraktionsebene (task level) dar (vgl. Kap. 2.4). Aus dieser abstrakten Beschreibung wird von BETTY eine Animation generiert, die als Präsentationsform das gestellte Präsentations- oder Visualisierungsziel erfüllt. Ein konkretes Beispiel hierzu: Das Visualisierungsziel

(localize-object "Knopf-1" 10)

erzeugt eine Animation, in der die Position des Objektes *„Knopf-1"* gezeigt wird, das heißt, das Objekt *lokalisiert* wird. Außerdem ist darin spezifiziert, daß die Präsentation dieser Information als Animation eine Dauer von 10 Sekunden haben soll.

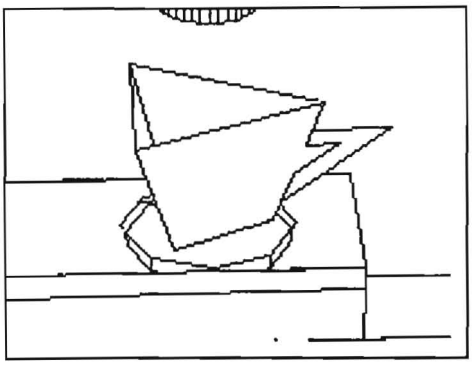
Andere notwendige Parameter zur konkreten Ausführung der Präsentation wie z.B. die Größe eines Ausgabefensters oder die Bildfrequenz werden als Parameter bei der Initialisierung des Planungsprozesses übergeben. Dabei handelt es sich jedoch um grundlegende Informationen, die mit den Inhaltsspezifikationen für die Animationen nichts zu tun haben und u.a. vom Layout vorgegeben werden.

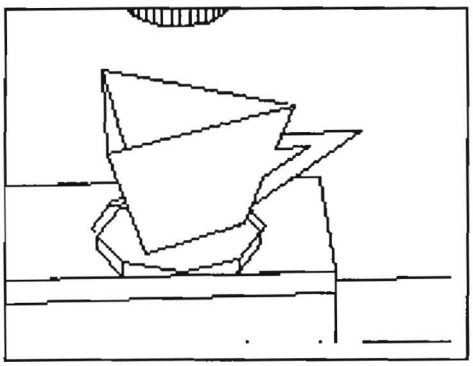
2.5.2 Spezifikation der Ausgabe

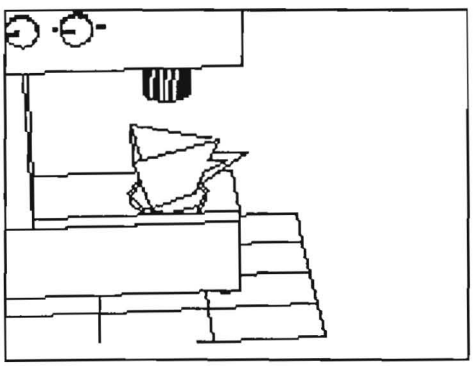
Die Ausgabe der Animationskomponente BETTY ist eine Folge von Einzelbildern, die, nacheinander in ein Fenster oder einen Bildschirmbereich kopiert, im zeitlichen Ablauf eine Animation bilden. Die Anzahl der Einzelbilder, aus denen die Animation besteht, ist abhängig von der Präsentationsdauer und der Bildfrequenz der Animation.

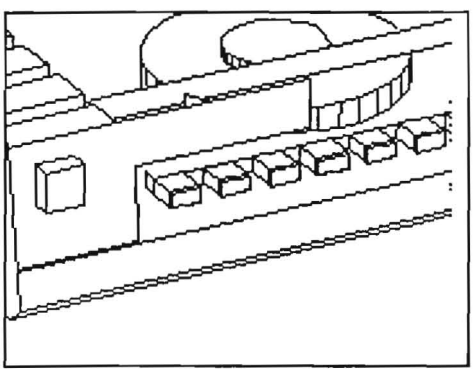
Diese Bildfolge wird zusammen mit einem Mechanismus zur Wiedergabe an die Layoutkomponente übergeben. In der Präsentation des gesamten Dokumentes wird dann zu gegebenem Zeitpunkt in einem gegebenen Bildschirmbereich Bild für Bild in schneller Folge abgespielt, so daß der Eindruck bewegter Objekte entsteht.

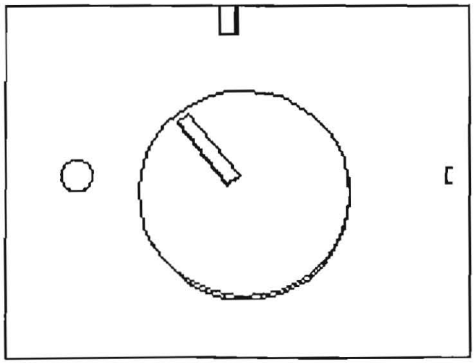
Haben alle Komponenten ihre Aufgaben erfüllt, so sollte der durch das Visualisierungsziel spezifizierte Effekt eintreten. Die Animation hat somit als Präsentationsform das ihr zugeteilte Präsentationsziel im gesamten Präsentationsplan erfüllt.











3 Analyse der Mittel

Nachdem nun klar ist, welche Aufgaben im einzelnen zu lösen sind, wollen wir uns einen kurzen Überblick über die theoretischen Grundlagen sowohl von informatischer als auch von filmtechnischer Seite her verschaffen. Hierbei soll immer die gestellte Aufgabe im Vordergrund stehen und die Verfahren jeweils auf ihre Tauglichkeit zu deren Lösung untersucht werden.

3.1 Zeitbeschreibungen

Um den Begriff *Zeit* zur Verwendung in Planungsverfahren oder anderen Prozessen mathematisch zu fassen, gibt es mehrere Ansätze, die die Zeit auf verschiedene Arten charakterisieren. So besteht einerseits die Möglichkeit, sich auf *Zeitpunkte* oder Situationen zu beziehen und bestimmte Ereignisse oder Zustände bestimmten Situationen zuzuordnen, andererseits ist es möglich, die Zeit immer auch als Dauer zu betrachten und bestimmten Ereignissen oder Zuständen bestimmte *Zeitintervalle* zuzuordnen.

3.1.1 Situationskalkül

Der erste Ansatz innerhalb der KI, die Größe *Zeit* formal zu fassen, wurde in [Mc69] vorgeschlagen und behandelt Zustandsbeschreibungen der Modellwelt zu bestimmten Zeitpunkten, die mit *Situationen* bezeichnet werden. Wollen wir beispielsweise bestimmten Aussagen über unsere Modellwelt Zeitpunkte oder Situationen zuordnen, so läßt sich das durch Prädikatenlogik ausdrücken, indem wir z.B. einem Prädikat der Art

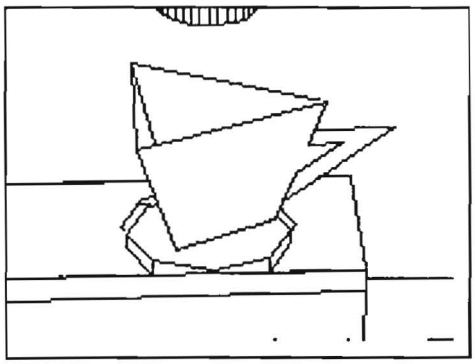
camera-points-to (X) oder *view-angle-set-for* (Y),

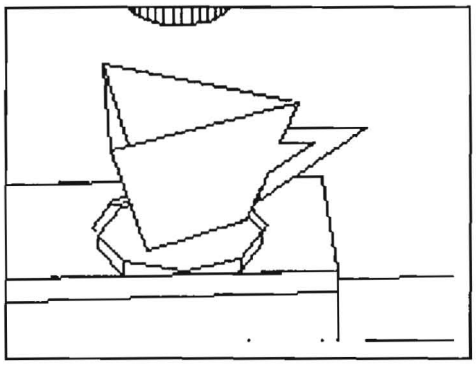
das jeweils einen Zustand der Modellwelt beschreibt (X und Y seien Objekte in dieser Welt.) eine Situation S_i zuordnen und sagen, daß das Prädikat in der Situation S_i gilt:

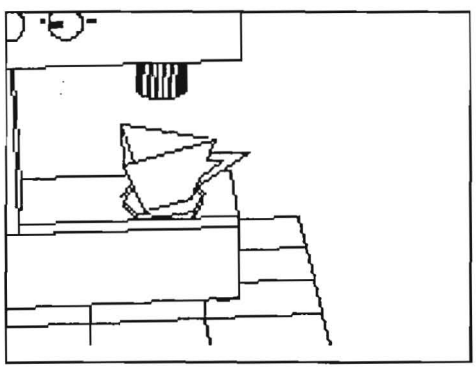
T *(aim-points-to* (X), S_1*)* bzw. T *(view-angle-set-for* (Y), S_2*)*.

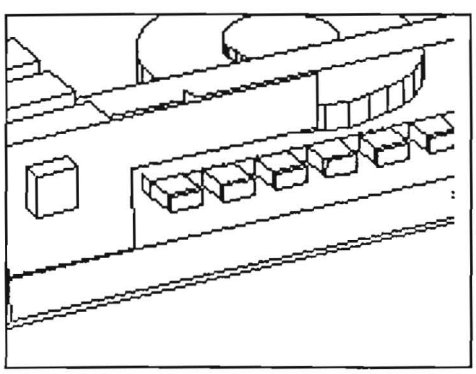
Die in unserer Welt ausführbaren Aktionen wie das Bewegen der Kamera und des Kamerazielpunktes oder die Bewegung von Objekten lassen sich dann ebenfalls durch Prädikate beschreiben:

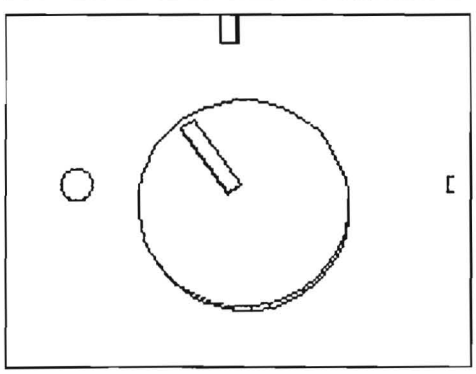
move-aimpoint-to (X) oder *move-object* (Y, M)











wobei X und Y Objekte der Modellwelt und M die Beschreibung einer Bewegung sind. Die Ausführung der Aktionen ergibt jeweils eine neue Situation:

do (move-aimpoint-to (X), S_3) bzw.
do (move-object (Y, M), S_4),

so daß wir über diese neuen Situationen wieder Aussagen machen können:

T (aim-points-to (X), (do (move-aimpoint-to (X), S)))

Für die Auswirkungen und Vorbedingungen der einzelnen Aktionen brauchen wir dann jeweils Axiome der Art

$$T\ (aim\text{-}points\text{-}to\ (X),\ S) \wedge Y \neq X \longrightarrow$$
$$T\ (aim\text{-}points\text{-}to\ (Y),\ (do\ (move\text{-}aimpoint\text{-}to\ (Y),\ S))) \wedge$$
$$\neg\ T\ (aim\text{-}points\text{-}to\ (X),\ (do\ (move\text{-}aimpoint\text{-}to\ (Y),\ S)))$$

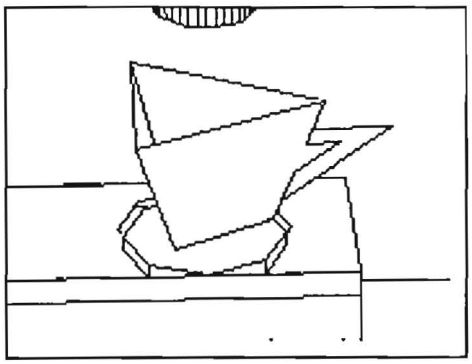
Durch eine solche Charakterisierung lassen sich die Zustände und Aktionen in der Modellwelt mittels Prädikatenlogik zwar zeitlich beschreiben, jedoch ist eine Aussage nur über ihre zeitliche Abfolge zu machen, nicht über ihre Dauer. Außerdem ist es nicht möglich, mehrere Aktionen gleichzeitig ablaufen zu lassen, was aber gerade eine grundlegende Forderung bei der Erstellung von Animationsdrehbüchern ist: Kamera- und Objektbewegungen können sehr wohl zeitlich parallel ablaufen und haben außerdem eine genau bestimmte Dauer.

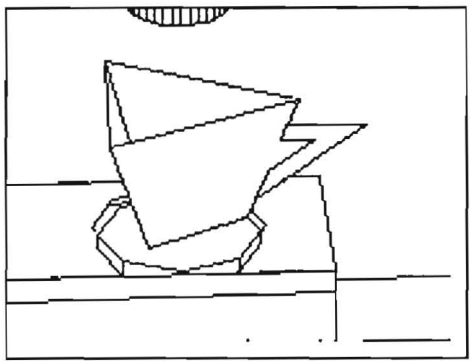
Die Charakterisierung der Zeit durch Situationen scheint also für die gestellte Aufgabe recht ungeeignet zu sein. Was wir jedoch aus dem Situationskalkül in die Praxis mitnehmen können, ist die Charakterisierung der Zustände in der Modellwelt, die in BETTYs Planer eine wesentliche Rolle als sogenannter Kontext spielen wird.

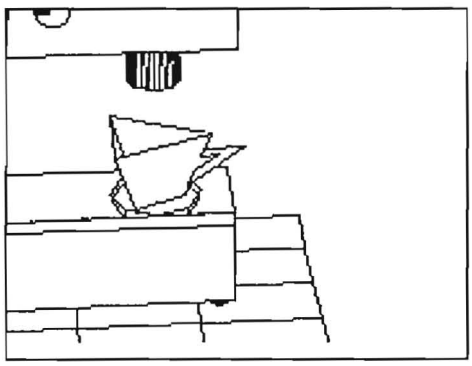
3.1.2 Intervalle

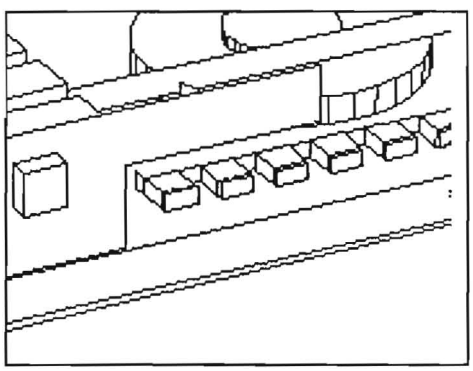
Wollen wir die Zeit in ihrer Dauer erfassen, so ist es sinnvoll, von Zeitintervallen zu sprechen. Diese Sichtweise wurde von Allen und Hayes in [Al85] vorgeschlagen und stellt eine mittlerweile weithin akzeptierte Beschreibung der Dimension Zeit dar. Jedem Zustand der Modellwelt und jeder Aktion wird ein Zeitintervall I als Paar von Zeitpunkten (a, b) zugeordnet, die jeweils Beginn und Ende bezeichnen. Das heißt, der Zustand Z dauert von Zeitpunkt a bis Zeitpunkt b, bzw. die Aktion A läuft von Zeitpunkt a bis Zeitpunkt b ab.

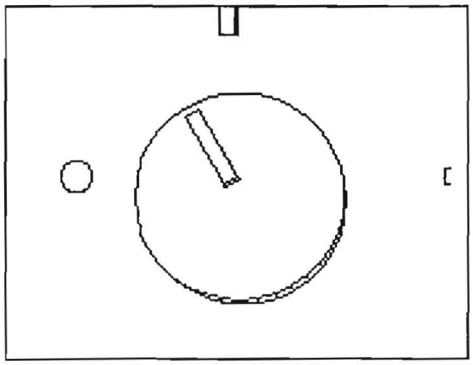
Solche Intervalle können sich auf verschiedene Art und Weise zueinander verhalten. Sie können sich überlappen, aneinander angrenzen, sich gegenseitig











enthalten oder ausschließen. Insgesamt gibt es 13 mögliche Relationen zwischen Zeitintervallen, die in Bild 6 anschaulich dargestellt sind.

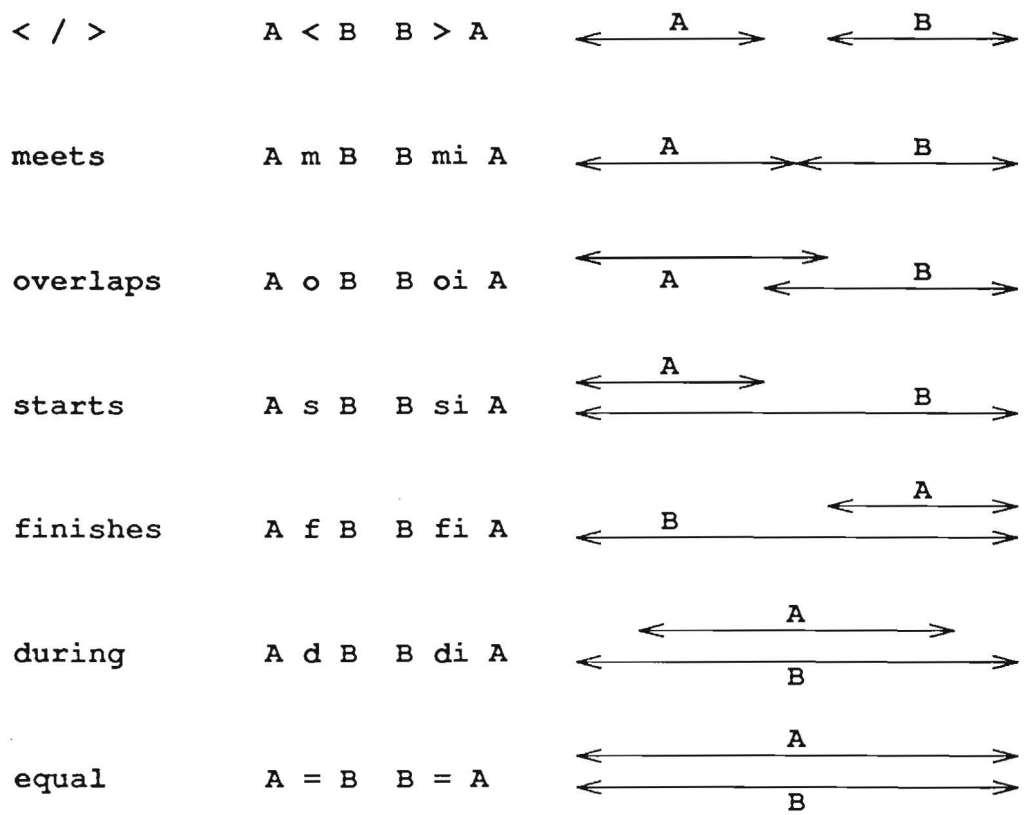
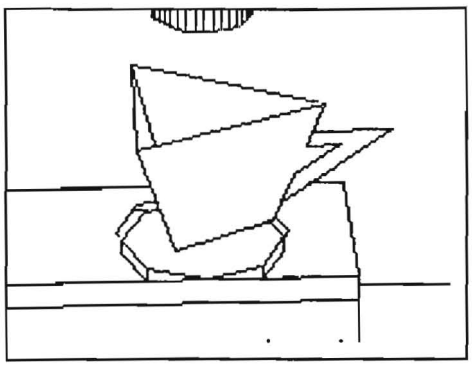
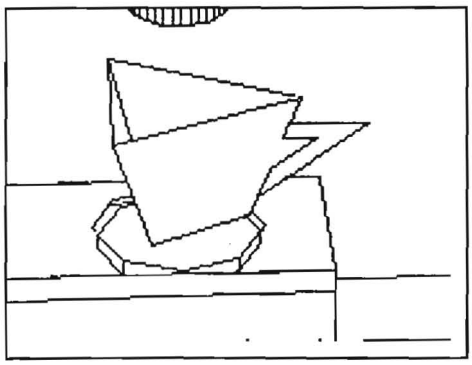


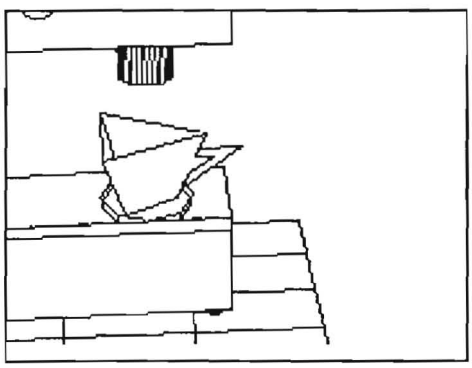
Abbildung 6: Die 13 möglichen Intervallrelationen

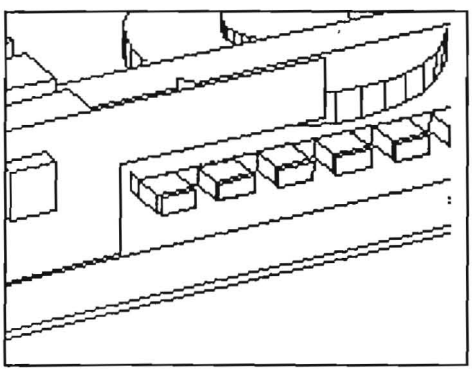
Auf diese Art und Weise lassen sich konkrete Aussagen über das Verhältnis zweier Zeitintervalle, also auch über das zeitliche Verhältnis zweier Aktionen oder Zustände, denen diese Intervalle zugeordnet sind, machen. Außerdem kann die Dauer von Aktionen direkt angegeben werden, sowie ihr Start- und Endzeitpunkt.

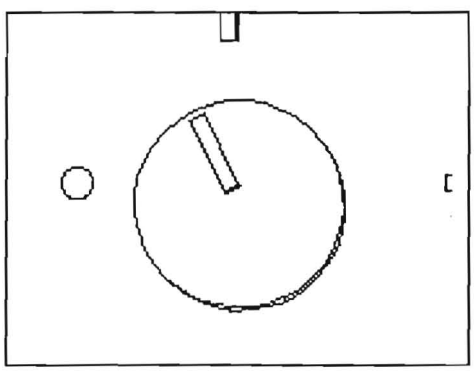
In einer Animation laufen nun mehrere Aktionen ab und bestehen verschiedene Zustände in bestimmten zeitlichen Verhältnissen zueinander. Zu jeder Aktion läßt sich Beginn und Ende auf einen genau bestimmten Zeitpunkt festlegen. Zu Beginn oder zu Ende einer Aktion ändert sich der Zustand der Modellwelt je nach den Auswirkungen der Aktion.











Baut man die Beschreibung einer Animation hierarchisch auf (siehe Kap. 2.4), so verlaufen die Zustände und Aktionen an den Blättern des Hierarchiebaumes stets zeitlich parallel (equal) oder aufeinanderfolgend (meets). Andere Zeitrelationen kommen in der Praxis in Animationen nur auf höheren Abstraktionsebenen vor und lassen sich dann auf niedrigeren Abstraktionsebenen durch die beiden Verhältnisse *Parallel* und *aufeinanderfolgend* umschreiben. Ein Beispiel hierfür ist in Bild 7 angegeben.

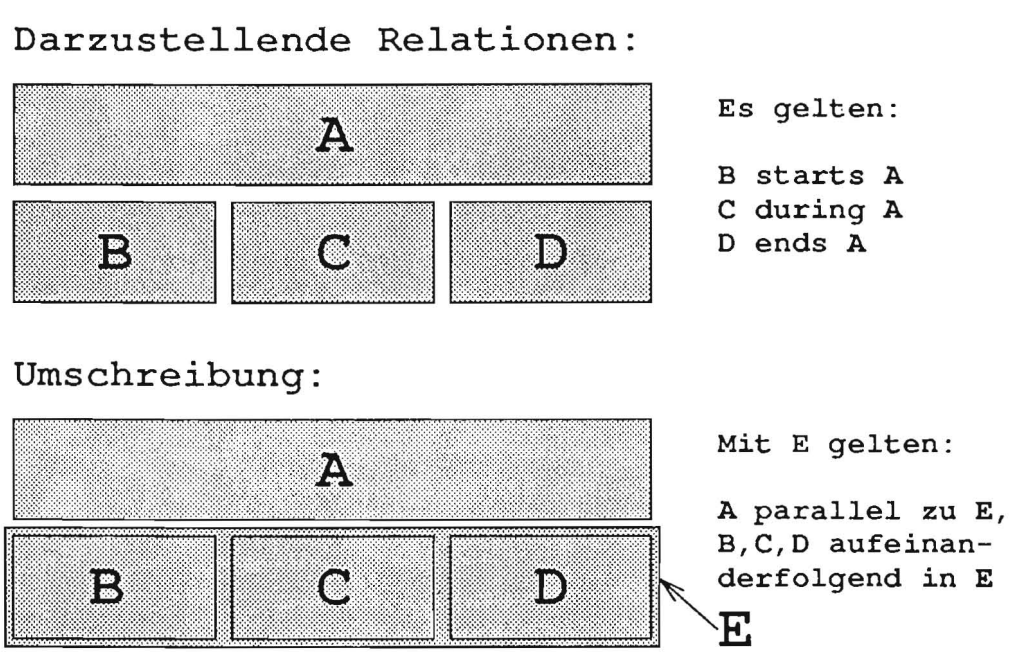
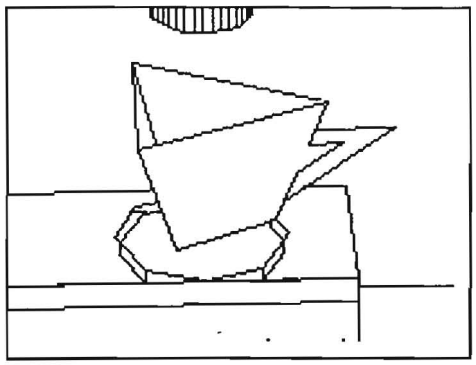
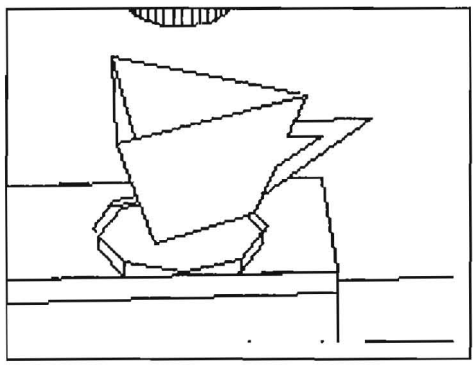
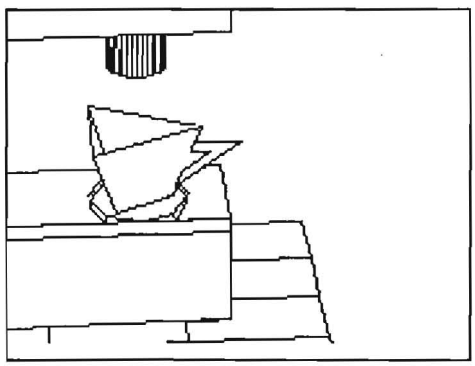


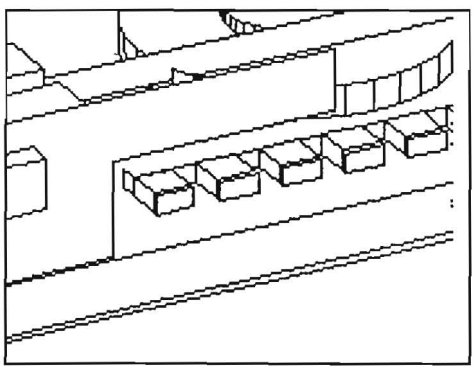
Abbildung 7: Umschreibung von Intervallrelationen

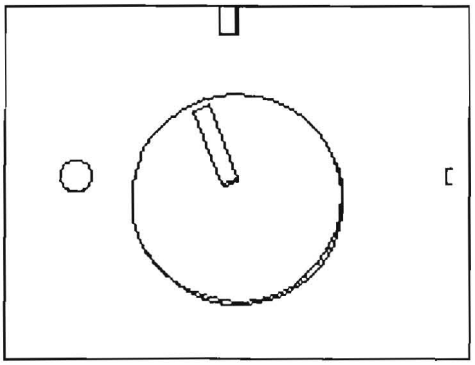
Die Intervallrelationen *starts*, *during* und *ends* werden hier durch die Konstruktion einer Hilfssequenz in eine Darstellung durch die Relationen *aufeinanderfolgend* und *parallel* überführt. Dies bedeutet, daß wir uns zur Planung von Animationen auf die entsprechenden Zeitrelationen *meets* und *equal* beschränken können, die sich in der aufeinanderfolgenden und parallelen Anordnung von Untersequenzen einer Animationssequenz ausdrücken. Diese Beschreibung der Dimension Zeit ist sehr gut geeignet, die bei der vorliegenden Aufgabenstellung auftretenden Probleme zu bewältigen, ohne dabei zusätzlich neue Probleme zu schaffen.











3.2 Planungsverfahren

Das Problem, eine Animation zu gestalten, kann, wie ganz zu Beginn gesagt, als Planungsproblem aufgefaßt werden. In der KI wurden verschiedene Planungsverfahren entwickelt und erforscht, von denen ich auf die beiden dem Problem nächstliegenden eingehen möchte.

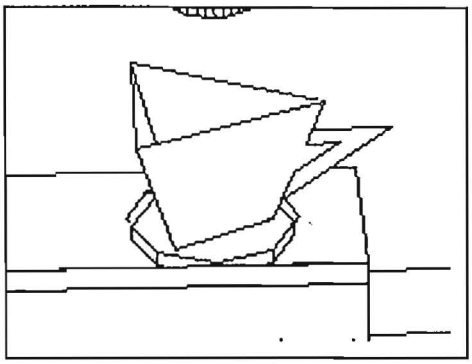
3.2.1 STRIPS

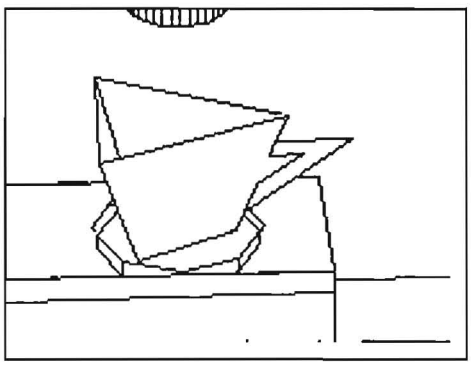
Wie wir bei der Formalisierung der Zeit und der Aktionen und Zustände in unserer Modellwelt gesehen haben, gibt es für die verschiedenen Aktionen bestimmte Vor- und Nachbedingungen. Zur Formalisierung dieses Sachverhaltes und zur Planung mit solchen Vor- und Nachbedingungen bietet sich der von Fikes und Nilsson in [Fi71] veröffentlichte STRIPS-Formalismus (STanford Research Institute Planning System) an. Darin wird ein Planoperator charakterisiert durch seine Vorbedingungen (Precondition), eine Menge von Aussagen, die nach Anwendung des Operators gelten (Add-Liste) und eine Menge von Aussagen, die danach nicht mehr gelten (Delete-Liste).

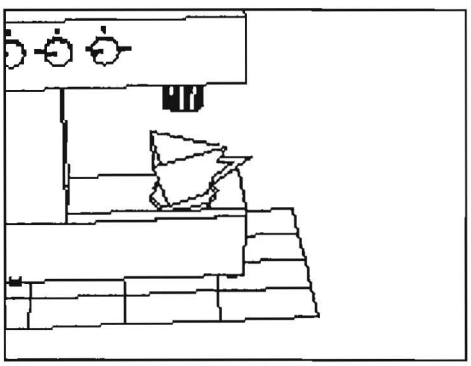
Operator: *move-aimpoint-to*
Precond.: *aim-points-to* $(X) \wedge Y \neq X$
Add: *aim-points-to* (Y)
Delete: *aim-points-to* (X)

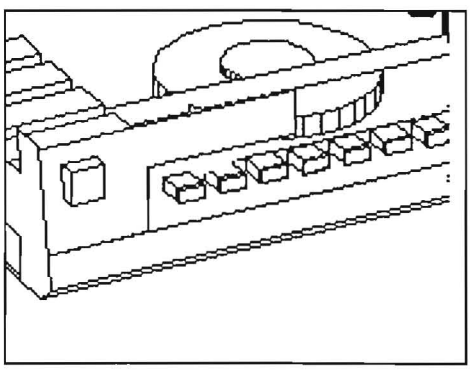
Mit einer Formalisierung aller in der Modellwelt möglichen elementaren Aktionen durch solche Planoperatoren sowie eine initiale Zustandsbeschreibung und eine Beschreibung des Zieles wäre nun der entsprechende Planungsalgorithmus anwendbar und würde eine Menge elementarer Aktionen samt partieller zeitlicher Anordnung liefern, die das gestellte Ziel erreichen. Der Algorithmus arbeitet einfach ausgedrückt etwa folgendermaßen:

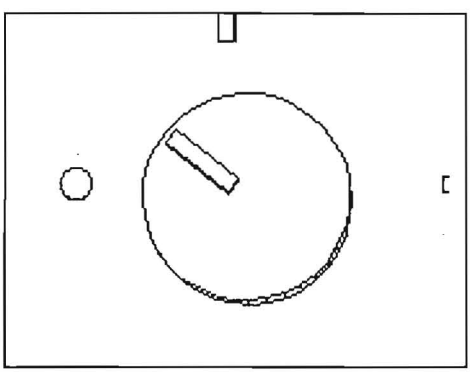
1. Gegeben eine initiale und eine gewünschte Welt M_0 und G_0
2. Versuche zu beweisen $M_i \vdash G_j$
3. Falls der Beweis fehlschlägt, suche Operatoren, die die fehlenden Bedingungen herstellen (mittels Add-Liste)
4. Die Vorbedingungen dieser Operatoren werden Unterziele G_j
5. Falls $M_i \vdash G_j$, dann wende Operator auf M_i an und gehe nach 2.











Der Suchraum eines solchen Planers wird jedoch schnell recht groß, da es zur Erreichung einer Vorbedingung oft mehrere Operatoren gibt. Somit müssen an dieser Stelle verschiedene Alternativen verfolgt werden.

Dieses Verfahren ist beim Finden unbekannter Lösungen und Pläne angemessen, da alle Lösungsmöglichkeiten in Betracht gezogen werden. Bei der Gestaltung einer Filmsequenz jedoch ist ein solcher Suchraum eher hinderlich. Durch welche filmtechnischen Mittel die Visualisierung eines Sachverhaltes oder einer Aktion zu erreichen ist, liegt oft klar auf der Hand und variiert lediglich in Abhängigkeit vom momentanen Zustand der Modellwelt. Die Zergliederung eines Visualisierungszieles kann also unter gewissen Nebenbedingungen ebensogut direkt angegeben werden, was eine Aufspaltung des Planungsvorganges in die Untersuchung mehrerer Alternativen verhindert.

3.2.2 Hierarchische Planung

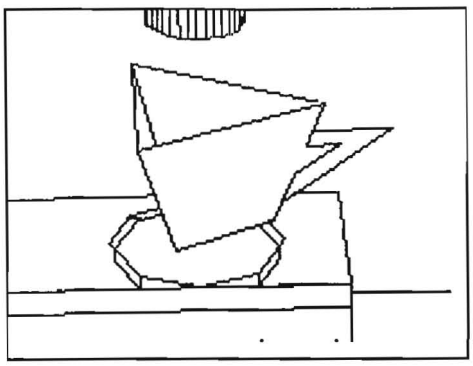
Diese Erkenntnis führt uns also zur hierarchischen Planung. Ein hierarchischer Planer der hier eingesetzten Art arbeitet mit Operatoren, die zu einem gegebenen Ziel die Menge der zu dessen Erreichung notwendigen Unterziele sofort angibt. Bei dem in BETTY verfolgten Planungsansatz besteht außerdem die Möglichkeit, entweder die zeitliche Anordnung dieser Unterziele anzugeben, oder nach bestimmten Bedingungen verschiedene Mengen von Unterzielen auszuwählen.

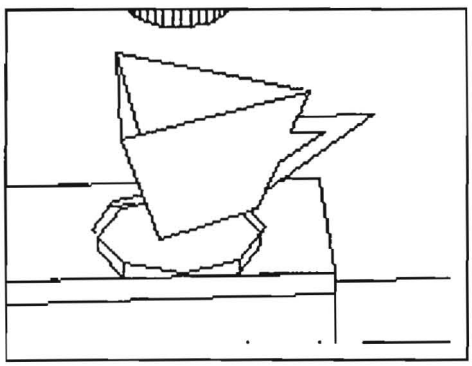
Wir erhalten also zwei Grundtypen von Planoperatoren. Der erste Typus bezeichnet Operatoren zur unbedingten Dekomposition eines Zieles, wie z.B.

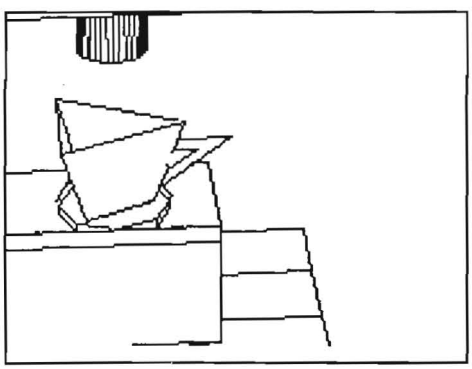
Operator: *zoom-to-object (X)*
Ordering: *parallel*
Subgoals: *(hold-cam-at-position*
hold-aim-at-position
adjust-va-for-object (X))

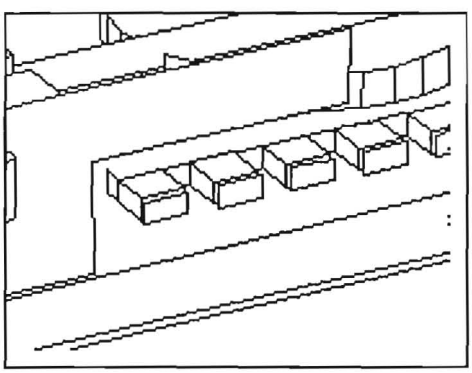
Die spezifizierten Unterziele stellen entweder elementare Aktionen in der Modellwelt dar oder es existieren wieder weitere Operatoren zu ihrer Auflösung. Die elementaren Aktionen haben außerdem bestimmte Auswirkungen auf den Zustand der Modellwelt, den Kontext, die im Detail in der Tabelle auf S. 41 angegeben sind.

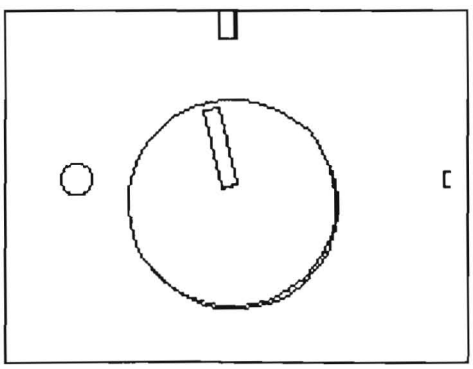
Somit sind die Auswirkungen eines Operators impliziert durch die Summe der Auswirkungen der von ihm implizierten elementaren Aktionen. Aufgrund von Bedingungen bezüglich dieses Zustandes der Modellwelt, des aktuellen Kontextes











also, bietet der zweite Operatortypus die Möglichkeit, zu verschiedenen Unterzielen zu verzweigen. Am Beispiel:

Operator: show-object-motion (X M)
if: *(∧ (camera-pos-set-for X)*
(aim-pos-set-for X))
then: *show-motion-var-1 (X M)*
if: *(∧ (camera-pos-set-for superobject (X))*
(aim-pos-set-for superobject (X))
(viewangle-set-for superobject (X)))
then: *show-motion-var-2 (X M)*
else: *show-motion-var-3 (X M)*

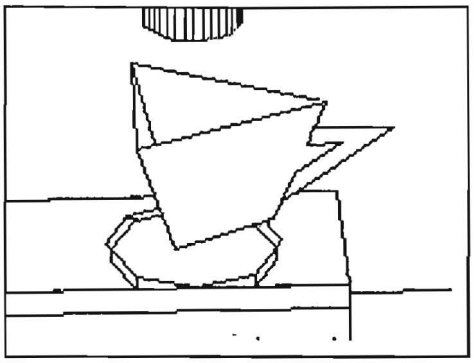
wobei *superobject* das jeweilige Oberobjekt eines Objektes (vgl. Kap. 4.7) bezeichnet. Mithilfe dieser beiden Typen von Planoperatoren ist es möglich, genau die erforderlichen Dekompositionen eines Visualisierungszieles in Abhängigkeit von seinem jeweiligen Kontext zu spezifizieren. Bei der konkreten Umsetzung des Verfahrens, die in Kap. 4.2 beschrieben ist, werden außerdem die jeweiligen Zeitintervalle, denen die Operatorinstanzen im konkreten Plan zugeordnet sind, bei der Instaziierung der Unterziele berücksichtigt und diese Unterziele neuen Zeitintervallen zugeordnet, die jeweils im Intervall des Oberziels enthalten sind. Wie diese Zuordnung geschieht, ist in jeder einzelnen Dekompositionsregel (Anhang A) genau festgelegt.

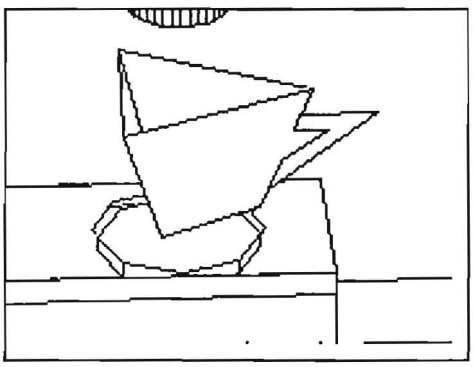
Auf diese Weise entsteht der Plan als hierarchische Struktur, als Baum, dessen Blätter durch elementare Aktionen in der Modellwelt gebildet werden und dessen Verzweigungen aus Dekompositionen eines Teilziels in zeitlich parallele oder aufeinanderfolgende Unterziele bestehen.

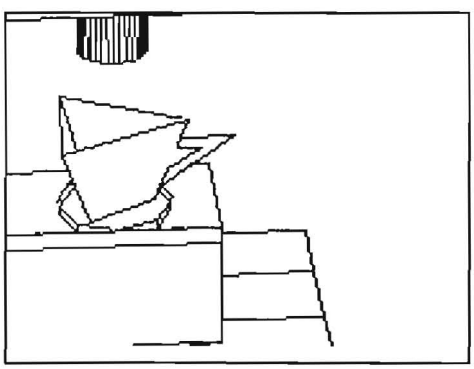
Diese Struktur des erzeugten Planes läßt sich nun exakt in eine hierarchische Beschreibung der geplanten Animationssequenz umsetzen, was ja schließlich auch unsere Absicht war. Der Plan wird ohne Aufblähung des Suchraumes direkt durch Entscheidung für eine Variante aufgrund des jeweiligen Kontextes aufgebaut, was ein sehr effizientes Verfahren zur flexiblen Generierung der gewünschten Datenstruktur bietet.

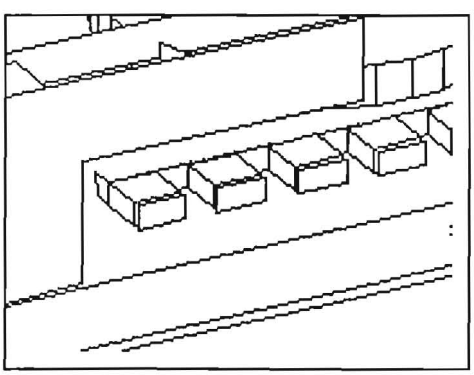
3.3 Grafische Verfahren

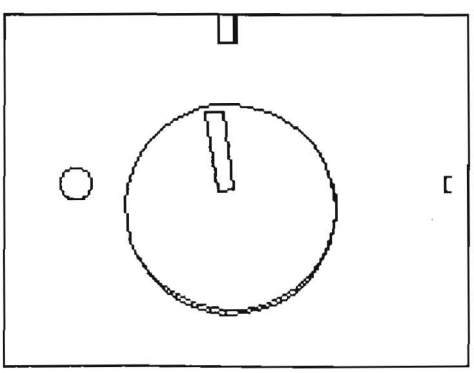
Neben einem geeigneten Verfahren zur Generierung eines Drehbuches für die Animation benötigt man natürlich auch grundlegende Methoden und Techniken zu ihrer Realisierung.











3.3.1 Modellierung und Darstellung

Objekte

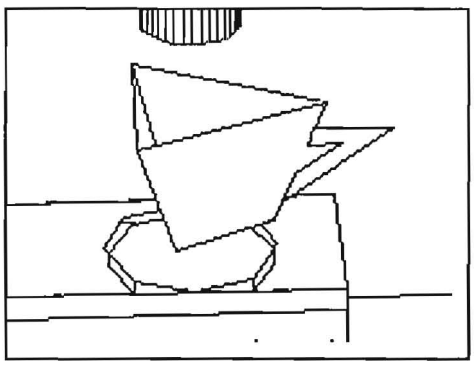
Zunächst einmal müssen die Modelle der in der Animation auftauchenden Objekte beschrieben werden. In WIP erfolgt diese Modellierung in S-Geometry, einem Drahtgitter-orientierten 3D-Grafiksystem. Die Objekte sind dort charakterisiert durch ihre Ecken, Kanten und Flächen, die in passenden Datenstrukturen abgelegt sind. In einer zusätzlichen Datenbasis werden außerdem einige weitere in WIP benötigte Angaben verwaltet, wie besonders ausgezeichnete Seiten der Objekte und umschreibende Quader zur effizienten Berechnung bestimmter räumlicher Informationen. Außerdem sind die Modelle entsprechend dem logischen Aufbau der jeweiligen Weltobjekte hierarchisch angeordnet. Es gibt eine Teil-von-Hierarchie, in der jedes Teilobjekt ein Oberobjekt und evtl. mehrere Unterobjekte hat.

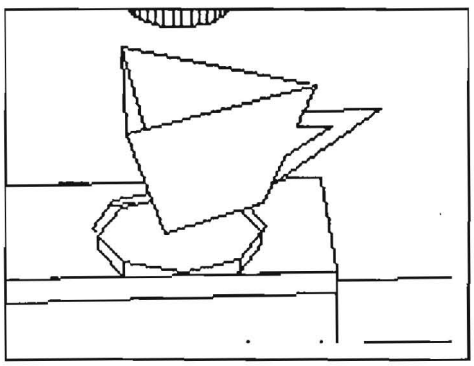
Dieser Ansatz ermöglicht die schnelle Darstellung der Objekte als Drahtgittermodell, stößt jedoch bei der Erzeugung von Szenerien mit Licht und Schatten auch an Grenzen: Kugeln und Zylinder sind z.B. nie richtig rund, da sie durch eine Zerlegung in ebene Flächen modelliert sind, und das in S-Geometry angewandte Lichtmodell weicht gravierend vom Verhalten des natürlichen Lichtes ab.

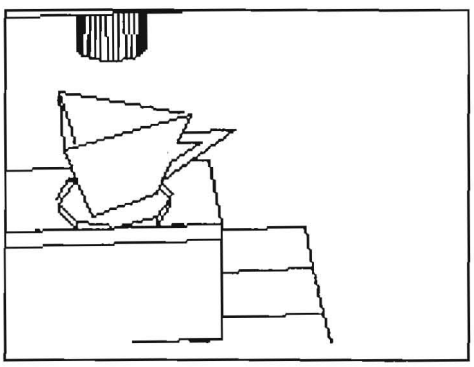
Dieses Beleuchtungsmodell und auch der enorme Anstieg des Rechenaufwandes bei der Berechnung realitätsnaher Bilder auf der vorhandenen Hardware legte es nahe, zur Erzeugung der Animationen bei der Darstellung der Objekte als Drahtgitter zu bleiben.

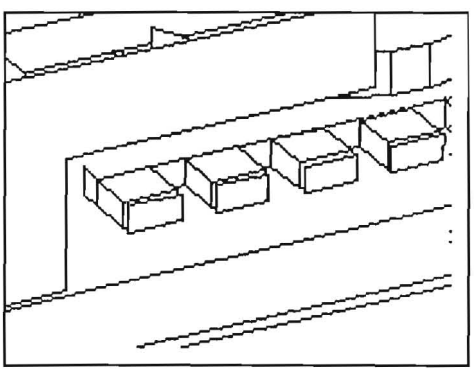
Bewegungen

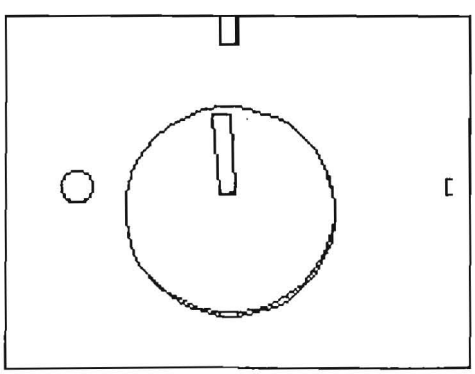
Desweiteren müssen die in der Animation vorkommenden Objektbewegungen modelliert werden. Dies geschieht in Form eines abstrakten Datentyps Bewegung, den die Grafikdesignkomponente (vgl. Kap. 4.7) zur Verfügung stellt. Bewegungen werden dort in drei verschiedene Grundtypen unterteilt, nämlich Rotation, Translation und Trajektorienbewegung. Dies steht in grundlegendem Gegensatz zu der in [Br94] beschriebenen Vorgehensweise, in der sich beliebig komplexe Bewegungen durch Verknüpfung elementarer Bewegungsformen wie Translation und Rotation beschreiben lassen. Letzten Endes muß aber auch dort eine Bewegung durch ihren Aufbau aus elementaren Bewegungen immer explizit angegeben werden. Die in WIP präsentierten Objektbewegungen werden hingegen als Trajektorien im Grafiksystem modelliert, wie auch die geometrischen Modelle der







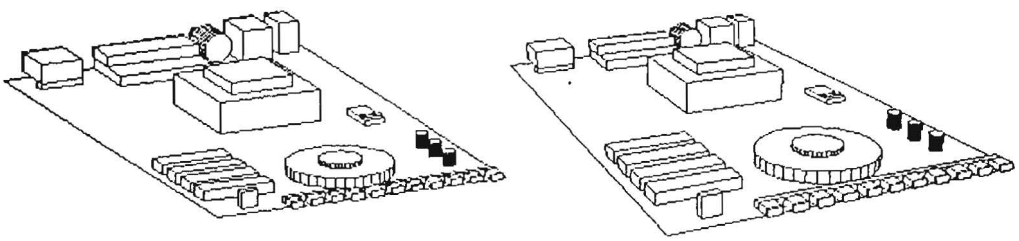




Objekte. Solche Trajektorien lassen sich beispielsweise mit dem System Geo-Antlima (vgl. [St93, Si94]) aus einer Menge raum-zeitlicher Propositionen ohne direkte Koordinatenangaben generieren. Würde ein Objekt beispielsweise von seiner derzeitigen Position auf ein anderes Objekt hinauf bewegt werden, so ließe sich die zugehörige Bewegungstrajektorie automatisch generieren, was eine weitere Einsparung an Modellierungsaufwand und eine größere Flexibilität des Systems zur Folge hätte.

3.3.2 Projektion und Perspektive

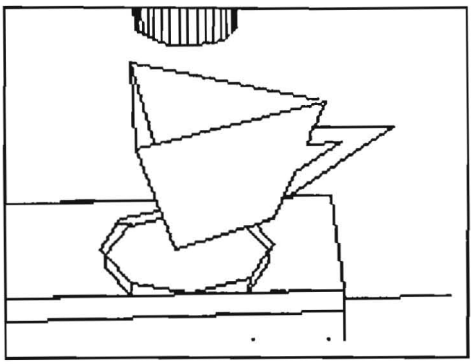
Zur Darstellung der dreidimensionalen Objekte am zweidimensionalen Bildschirm muß eine Projektion der dritten Raumdimension auf die am Bildschirm verfügbaren zwei Flächendimensionen erfolgen. Standardverfahren hierzu sind die Zentralprojektion, die der Abbildung durch ein fotografisches Objektiv entspricht ([Ma88]), und die Parallel- oder Orthografische Projektion, die mathematisch einfacher aber optisch unrealistischer ist.

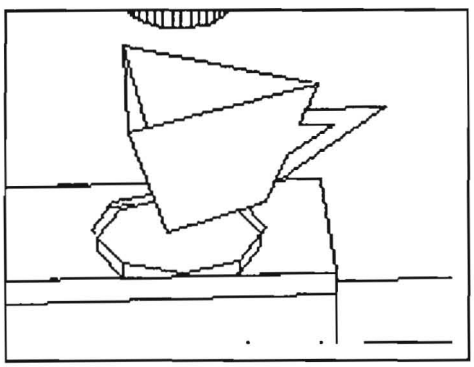


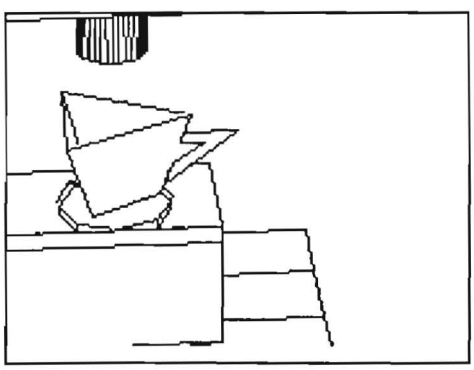
Orthografische Projektion Zentralprojektion

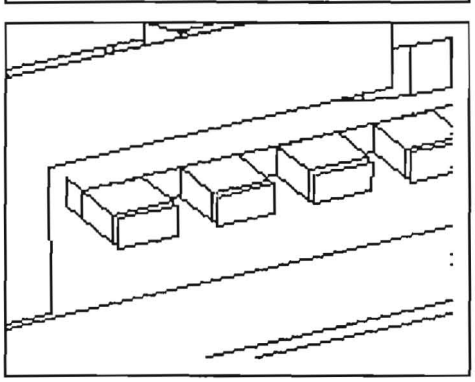
Abbildung 8: Verschiedene Möglichkeiten der Projektion

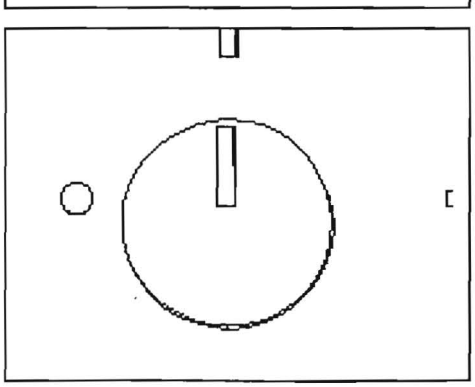
Bei der Zentralprojektion verlaufen alle zur Bildebene parallelen Geraden auch im Bild parallel und alle zur Bildebene senkrechten Geraden treffen sich in einem Fluchtpunkt. Die Lage dieses Fluchtpunktes ändert sich mit der Brennweite des Objektivs und wandert bei sehr langen Brennweiten ins unendliche, was bedeutet, daß sich das Bild dann nicht mehr von dem nach der Parallelperspektive unterscheidet. Bei der Parallelperspektive laufen alle im Dreidimensionalen parallelen Geraden auch im zweidimensionalen Abbild parallel. Die Parallelprojektion entspricht einer Zentralprojektion aus unendlicher Entfernung mit unendlicher Brennweite des Objektivs.











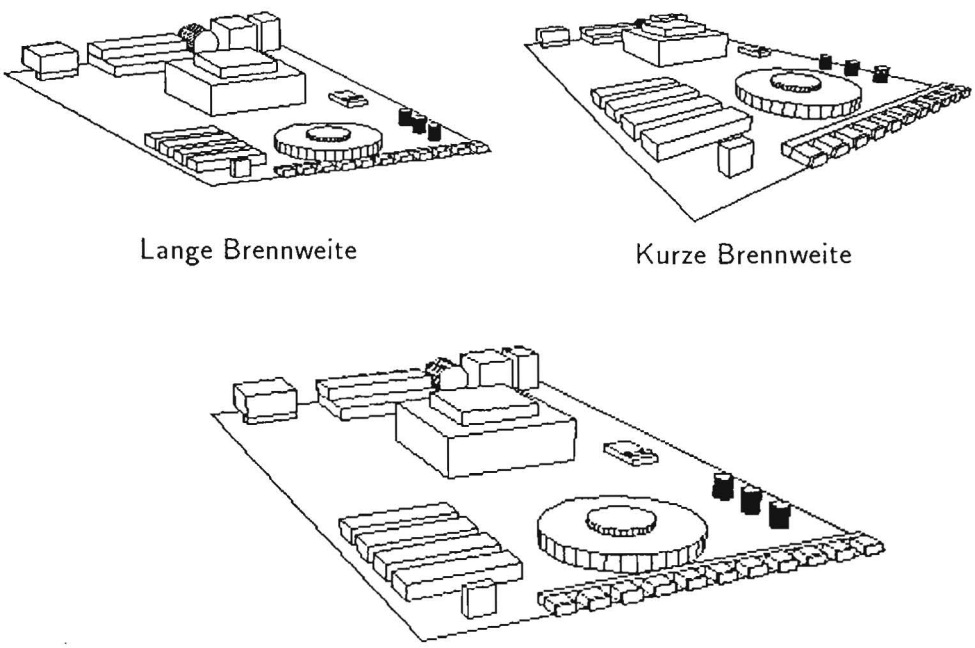
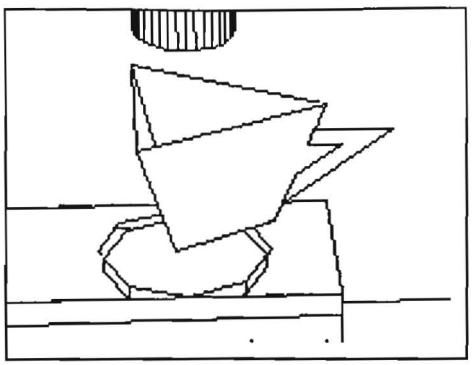
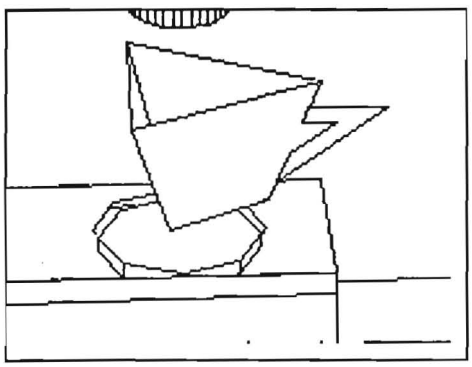


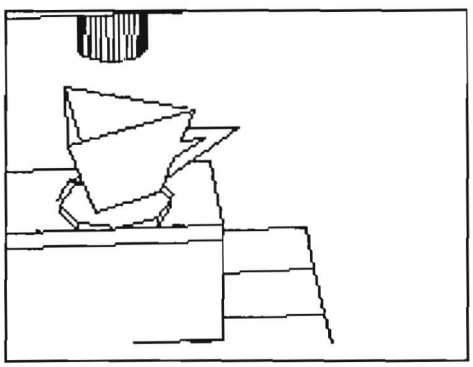
Abbildung 9: Wirkung verschiedener Brennweiten

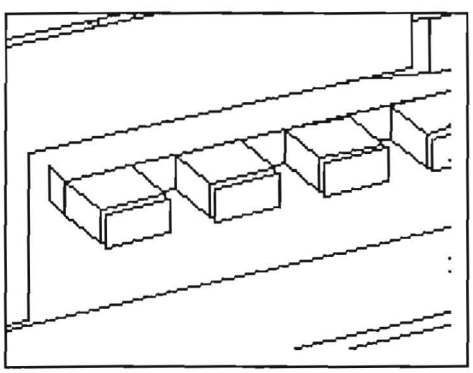
Der Unterschied der Bildwirkungen zwischen einer Nahaufnahme mit einem Weitwinkelobjektiv und einer weiter entfernten Aufnahme mit einem Teleobjektiv kann die Bildaussage gezielt unterstützen, aber auch verfälschen, sobald dieses Mittel unbedacht eingesetzt wird. Objekte, die von Nahem mit sehr kurzer Brennweite aufgenommen werden, wirken übertrieben perspektivisch und dadurch oft sehr dramatisch oder einfach groß, Objekte, die aus großer Entfernung mit einer langen Brennweite aufgenommen werden, wirken hingegen eher sachlich, distanziert (siehe auch [Ma88]).

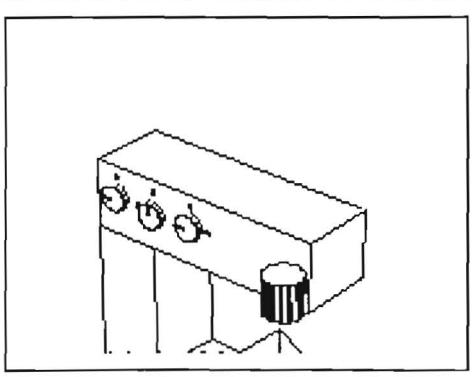
Um diese als Seiteneffekt auftretende Semantik auszuschließen, wird in BETTY sichergestellt, daß Standardeinstellungen mit einer Normalbrennweite, das heißt einer Brennweite, deren Bildwinkel etwa dem Betrachtungswinkel der Abbildung entspricht, aufgenommen werden. So entsteht ein neutraler Bildeindruck, der den Betrachter in Beziehung zum entstehenden Bild setzt, jedoch nicht von der eigentlichen Bildaussage ablenkt. Die verschiedenen Bildwirkungen sind aus Bild 9 zu ersehen.











3.4 Filmtechnische Verfahren

Die Produktion eines Filmes findet in mehreren Stufen statt: Zuerst werden alle einzelnen Einstellungen des Filmes gedreht, meist mehrmals, und oft unter Verwendung verschiedener Bildausschnitte und Kamerapositionen zur Erzielung verschiedener Bildwirkungen. In einem nachfolgenden Arbeitsgang werden die einzelnen Einstellungen zu einem zusammenhängenden Film verbunden (Schnitt).

Ein Standardverfahren ist z.B., in jeder Einstellung einmal die Szenerie im Gesamten aufzunehmen, um einen kompletten Überblick vermitteln zu können (long shot), einmal die Szenerie aus mittlerem Abstand mit einer mittleren Menge an sichtbaren Details (medium shot) und einmal von ganz nahe, um alle Details der Handlung zu zeigen (close shot). Dieses Verfahren (triple take filming genannt,) erlaubt es, nachher beim Schnitt des gesamten Films eine ausgewogene Mischung aus Übersichtsaufnahmen und Details zu erreichen, um dem Zuschauer diese verschiedenen Ebenen gleichzeitig zu vermitteln.

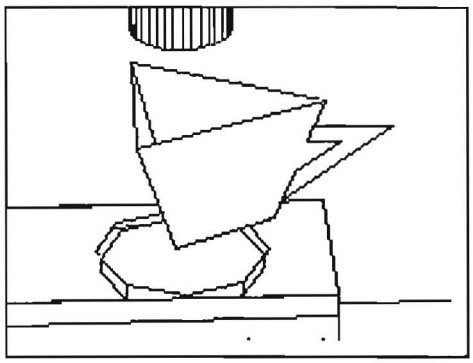
Bei der Generierung einer Animation fallen diese beiden Arbeitsgänge zusammen: keine Szene wird mehrfach gedreht, sondern nur einmal mit den vorher festgelegten Kameraparametern und -Positionen berechnet.

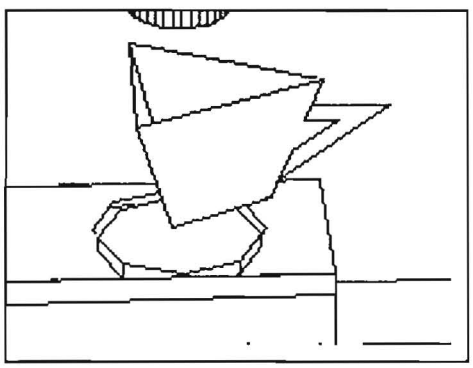
3.4.1 Schnitt

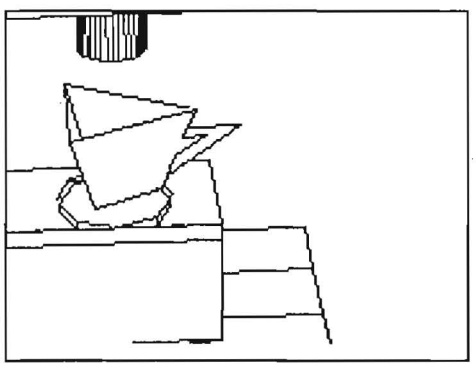
Das, was im Film durch den Schnitt erreicht wird, nämlich die plötzliche Veränderung der Szenerie durch die Verbindung verschiedener Filmstreifen aus verschiedenen Einstellungen, geschieht in der Animation durch das abrupte Verändern der Kameraeinstellungen oder der Szene selbst. Diese Zeitpunkte sind oft Eckpunkte im Drehbuch der Animation, an denen bestimmte Standardpositionen und Parameter neu eingestellt werden oder Objektbewegungen ihren Start- oder Endzustand erreichen, und deren Einstellungen auch als Schlüsselbilder im Storyboard auftauchen. Typischerweise geht eine Sequenz unterster Ebene eines hierarchisch aufgebauten Animationsscripts von einem solchen Schnitt bis zum nächsten. Ausnahmen sind mehrteilige Bewegungen und umfangreiche mehrteilige Kamerafahrten während einer einteiligen Objektbewegung.

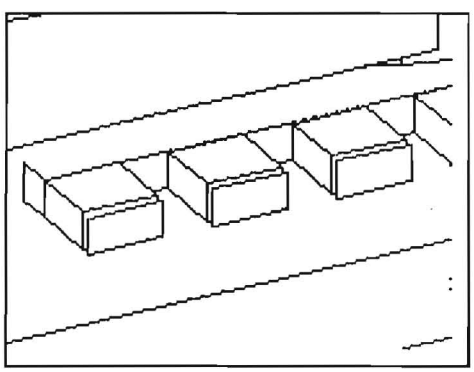
Sonderformen des Schnittes werden durch das Ein- und Ausblenden des Bildes (nach Schwarz oder Weiß) realisiert. Überlagert man dies zeitlich, so läßt sich so sogar eine Szene in eine andere überblenden, was in Filmen oftmals das Vergehen einer langen Zeit oder das Überspringen großer Entfernungen symbolisiert.

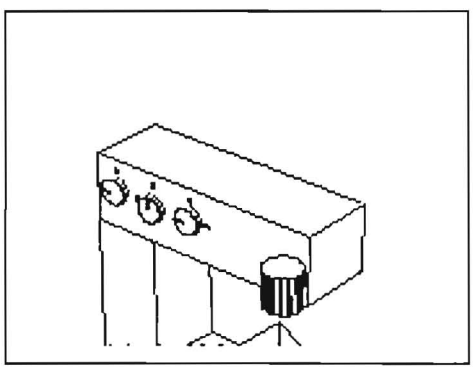
Alles in allem verliert der Schnitt seine direkte Funktion aus dem Film, da die Animation in einem Stück berechnet wird und kein Schneiden notwendig macht.











Trotzdem ist der Schnitt als filmtechnisches Mittel mit Semantiken belegt, die er auch in der Animation behält. Schnitte werden deshalb in Animationen trotzdem weiterverwandt, realisiert durch eine plötzliche Veränderung der Kameraposition oder -Einstellungen oder der Szenerie von einem Bild zum nächsten.

3.4.2 Kamerafahrt und Kameraschwenk

Soll die Kameraposition verändert werden, während eine Handlung abläuft, so ist es sehr verwirrend, dies durch einen Schnitt zu realisieren. Der optische Fluß der Animation wird gestört, und der Betrachter muß sich zuerst neu orientieren, um die Handlung vom neuen Standpunkt aus mitverfolgen zu können.

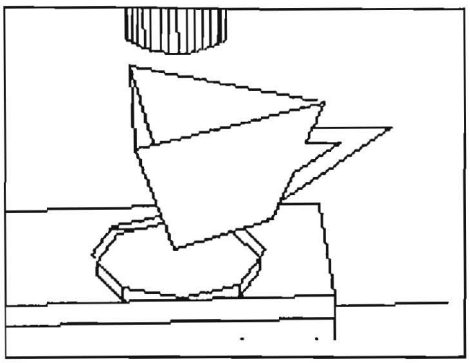
Dies läßt sich durch eine Kamerafahrt verhindern: Die Position der Kamera verändert sich allmählich über einen gewissen Zeitraum hinweg, um in einer neuen Position wieder statisch zu werden. Der Betrachter nimmt die Bewegung der Kamera selbst oft kaum wahr und behält zu jedem Zeitpunkt die Orientierung. Darüberhinaus hat eine Kamerafahrt sogar noch den Nutzeffekt, daß durch ein Zeigen der Szenerie von verschiedenen Positionen aus ein wesentlich plastischeres Bild vermittelt wird, als dies mit einer statischen Kameraposition möglich ist (siehe hierzu auch Kap. 2.2.1). Die Kamerafahrt bietet sich also vor allem an, um räumliche Zusammenhänge sichtbar zu machen.

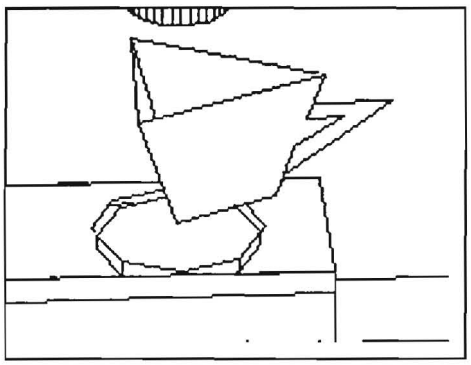
Ein Kameraschwenk hingegen läßt die Position der Kamera selbst fest und verändert nur deren Zielpunkt. Somit wird das Verfolgen sich bewegender Objekte möglich und genau wie bei der Kamerafahrt wird der Betrachter nicht durch einen abrupten Wechsel der Szenerie verwirrt. Die beiden Verfahren lassen sich natürlich beliebig kombinieren sowie mit der Veränderung des dritten Kameraparameters, der Brennweite verbinden.

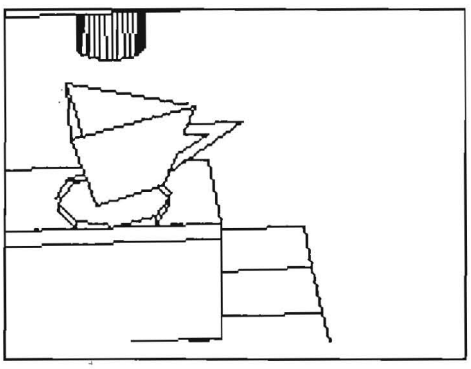
3.4.3 Zoom

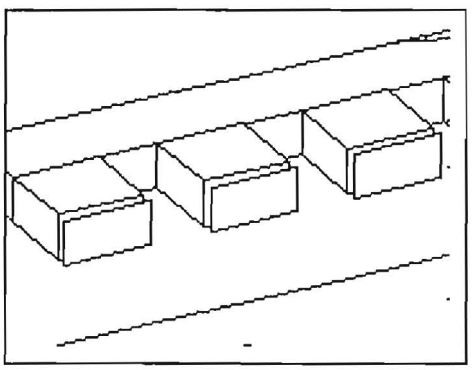
Das Verändern der Kamerabrennweite bei laufendem Bild wird mit zoom bezeichnet. Es ermöglicht die Regulierung des Abbildungsmaßstabes und somit des Bildausschnittes unter Beibehaltung der Kameraposition. Dieses Mittel bietet sich hervorragend dazu an, von einer Übersichtsaufnahme der Szenerie langsam bis zu einer Detailaufnahme eines Objektes zu wechseln, wobei der gesamte visuelle Kontext bewahrt bleibt.

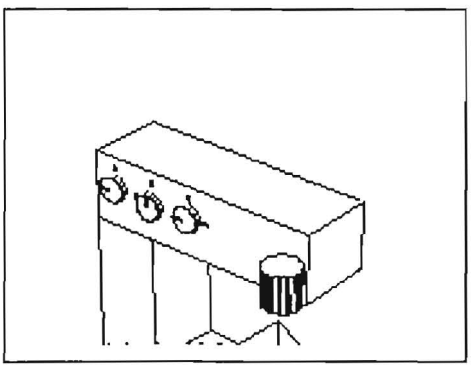
Obwohl zu Ende der Einstellung nur noch ein Detail im Bild sichtbar ist, verbleibt beim Betrachter eine Vorstellung über den gesamten räumlichen Zusammenhang der Szenerie, was ähnlich dem triple take filming zu einer gleich-





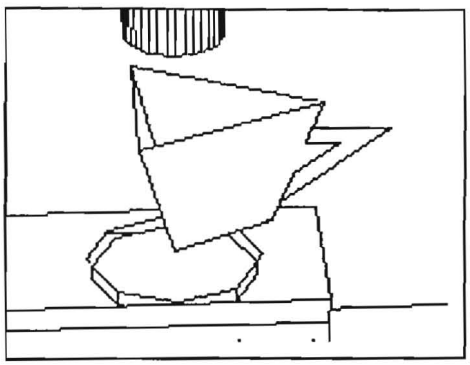


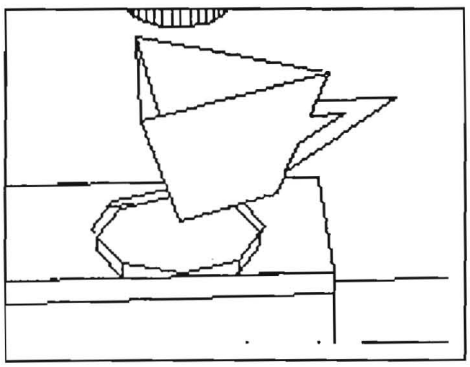


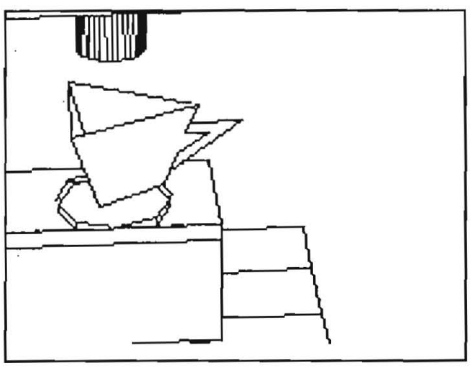


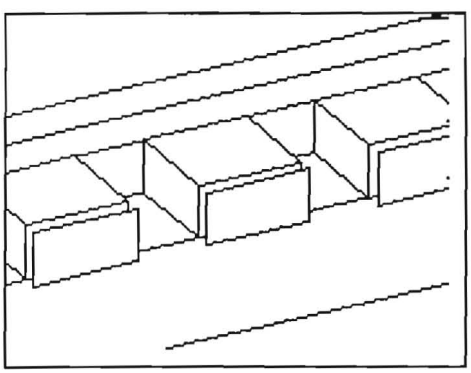
zeitigen Übermittlung verschiedener Komplexitätsebenen dient. Umgekehrt läßt sich natürlich durch einen Wechsel vom engen Ausschnitt zum Gesamten die Einordnung eines Details in seine Umgebung direkt visualisieren. Dies ist beispielsweise anwendbar, wenn zwei Teile, die sich im gleichen visuellen Kontext befinden, zueinander in Beziehung gesetzt werden sollen.

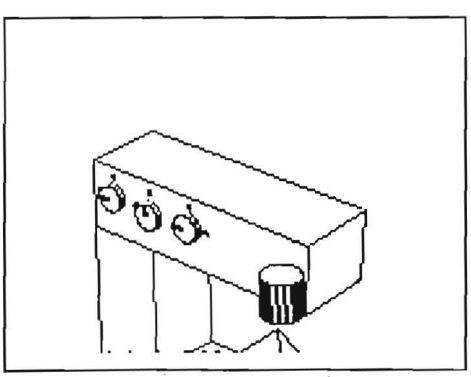
Die Wirkung dieser Verfahren ist im Daumenkino Nr. 4 zu ersehen, in dem eine Lokalisation der beiden in der Frontleiste befindlichen LEDs allein durch zwei Zooms und einen kleinen Kameraschwenk erreicht wird.











4 Die Arbeitsweise des BETTY-Systems

4.1 Gesamtstruktur des Systems

Die Architektur des BETTY-Systems innerhalb von WIP ist schematisch in Bild 10 dargestellt. Links befindet sich dabei der eigentliche Animationsplaner, der während der Planung auf die Toolbox TOPAS zugreift und das geplante Script an die Animationsrealisation auf der rechten Seite weitergibt.

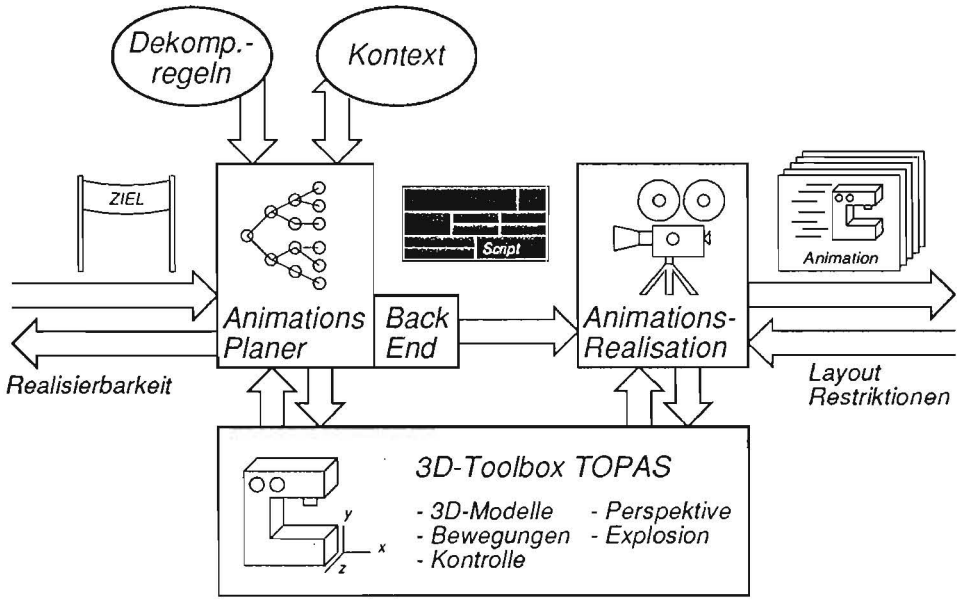
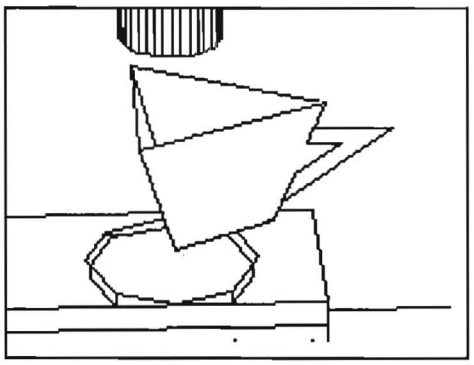
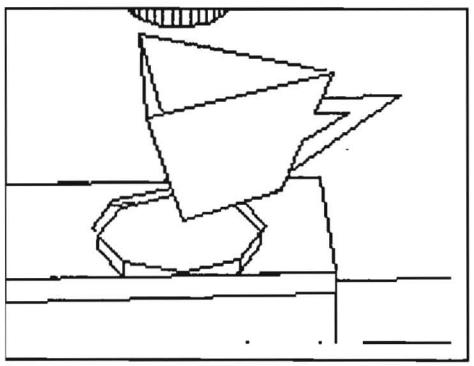


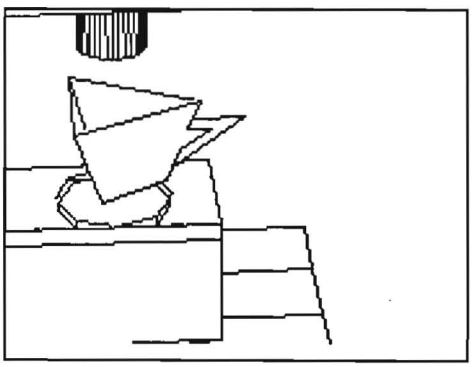
Abbildung 10: BETTY-Gesamtstruktur

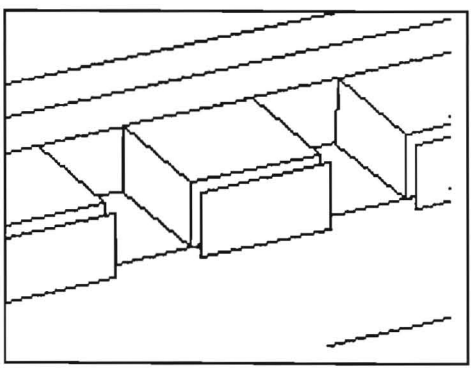
Wie zu Beginn der Arbeit schon beschrieben, weist in WIP ein zentraler Präsentationsplaner die darzustellenden Informationen den Planungs- und Realisierungskaskaden der einzelnen Ausgabemodi zu. Parallel zu BETTY existieren (siehe Bild 5 auf S.20) die Komponenten zur Planung und Realisierung natürlicher Sprache (Text) sowie zur automatischen Erstellung technischer Grafiken. Näher beschrieben sind diese Systemteile in [Sc94, Ha91, Ri92, Sc92].

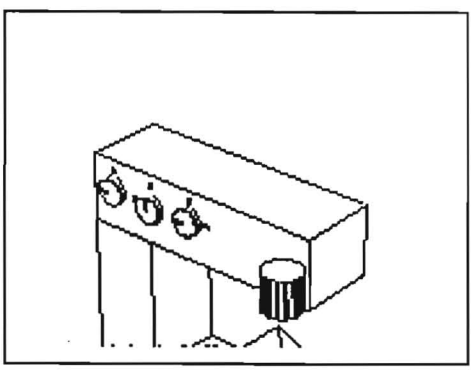
Ein Visualisierungsziel, wie es in Abschnitt 2.5.1 beschrieben ist, bildet die Eingabe der Animationskomponente. Der Animationsplaner zergliedert dieses Ziel mithilfe einer Menge von Dekompositionsregeln unter Berücksichtigung eines











Kontextes in Unterziele, die auf unterster Ebene durch elementare Aktionen der Kamera oder der Objekte realisiert werden. Der so erstellte Plan für die Animation wird durch eine Schnittstellenkomponente (Backend) in die vom Animationssystem benötigte Datenstruktur übersetzt, die dort mit Script bezeichnet wird.

Diese Schnittstelle zwischen Planungs- und Realisierungskomponente übernimmt außerdem den Austausch geometrischer Daten, die während des Planungsprozesses benötigt oder erzeugt werden, soweit dies nicht durch TOPAS erledigt wird. Das birgt in sich den weiteren Vorteil, daß bei einem Wechsel der Ausgabekomponente (z.B. zur Erreichung einer besseren oder schnelleren Ausgabe) nur diese Schnittstelle geändert werden muß, während der Planer selbst von der Änderung nicht betroffen ist.

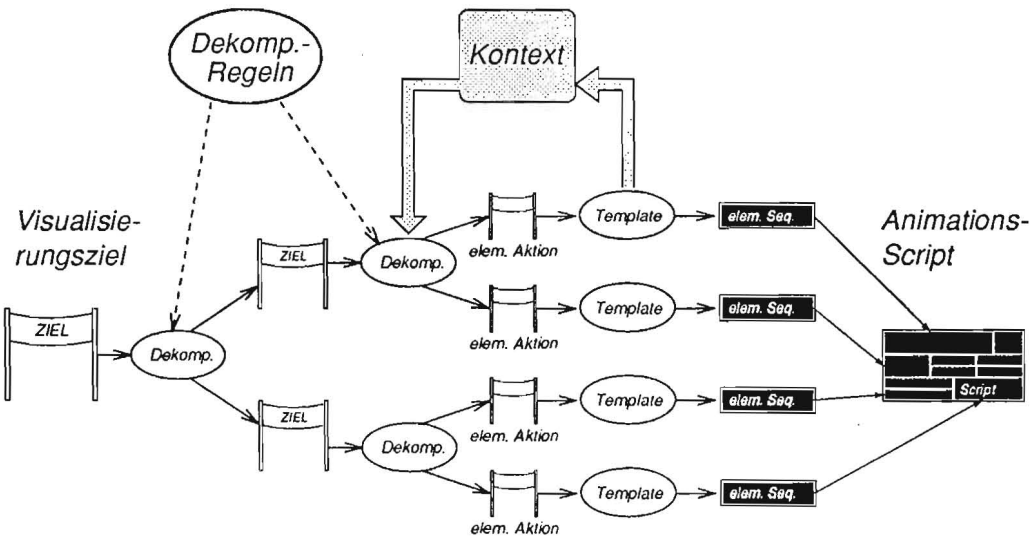
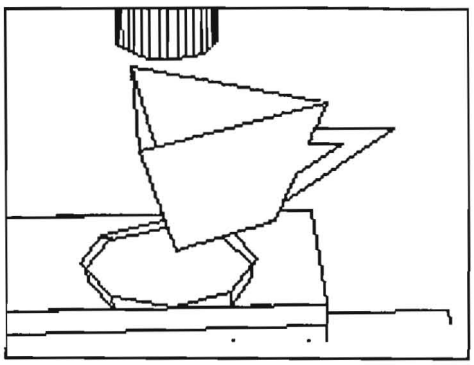
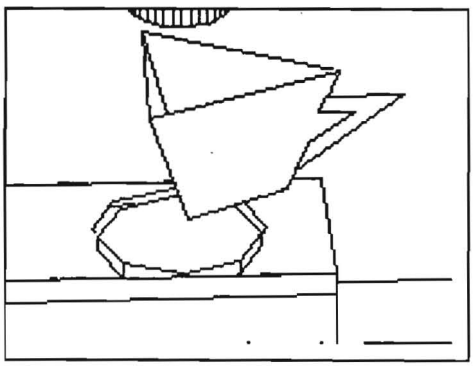
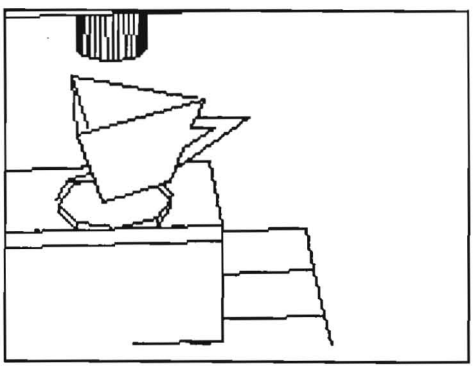


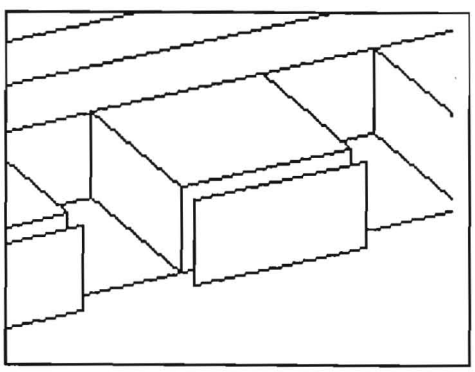
Abbildung 11: Dekomposition eines Visualisierungsziels und Übersetzung in elementare Scriptsequenzen

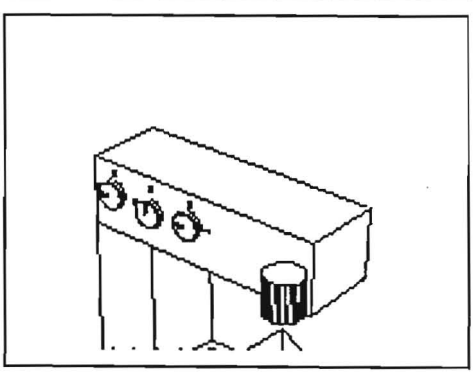
Die Animationsrealisierung schließlich wird komplett von einem kommerziellen Grafikpaket übernommen, natürlich nach den Vorgaben und unter der Kontrolle der Planungskomponente. Da das Animationsscript inkrementell erzeugt wird, kann diese Berechnung bereits mit der Generierung der ersten elementaren Scriptsequenz beginnen, so daß in gewissem Sinne eine Realzeit-Animation möglich ist. Ausgabe des Grafikpaketes ist eine Animation in Form einer Folge von Bitmaps, die durch vorhandene Mechanismen auf beliebige Bildschirmteile wiedergegeben











werden können. Ort und Zeitpunkt dieser Wiedergabe aber sind wiederum Sache der Layoutkomponente, die die Animation außerdem mit den Ausgaben der anderen Komponenten koordinieren muß. Die Funktion der einzelnen Teile wollen wir uns nun im Detail ansehen.

4.2 Die Planungskomponente

Die Animationsplanungskomponente, die derzeit in BETTY die Planung aller Kamera- und Objektbewegungen übernimmt, realisiert ein kontextgesteuertes hierarchisches Planungsverfahren. Zu jedem Ziel- oder Unterzieltypus existiert entweder eine Dekompositionsregel, die angibt, wie es weiter zu zergliedern ist, oder eine Realisierung als elementare Aktion auf Ebene des Scriptes. Der genaue Algorithmus des Planers wird in Abschnitt 4.2.5 beschrieben, nachdem die zum Verständnis notwendigen Begriffe eingeführt sind.

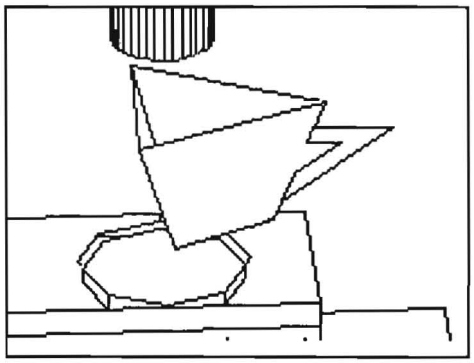
4.2.1 Elementare Aktionen

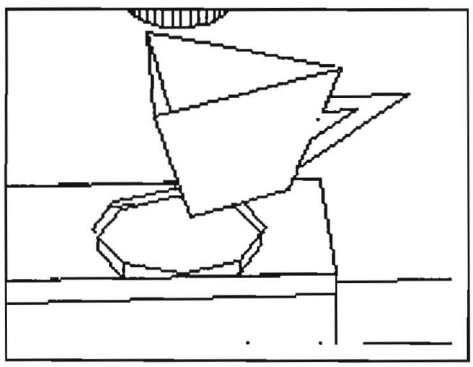
Die benötigten elementaren Aktionen sind Bewegungen oder Veränderungen von Eigenschaften eines Objektes oder der Kamera innerhalb einer bestimmten Zeitspanne um einen bestimmten Betrag oder entlang einer bestimmten Achse oder Trajektorie. Am konkreten Beispiel:

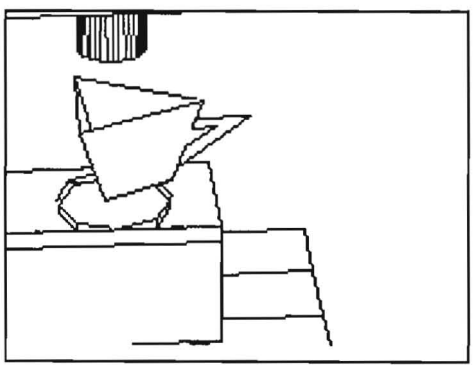
> *„Kamerafahrt vom derzeitigen Punkt aus in die Position für Objekt X innerhalb 2 Sekunden"*
>
> *„Kamerazoom von der aktuellen Brennweite zu einer bildfüllenden Darstellung für Objekt Y innerhalb 1.5 Sekunden"*
>
> *„Bewegung des Objekts Z entlang Trajektorie T innerhalb 5 Sekunden"*

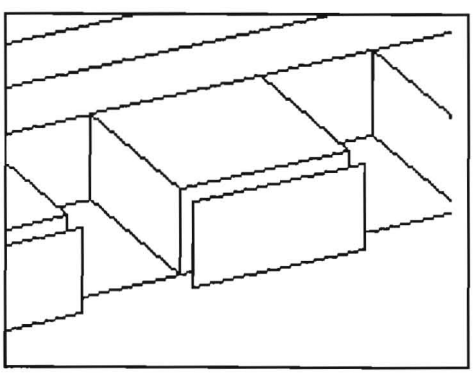
Insgesamt existieren 10 Typen solcher elementarer Aktionen, nämlich 7 Kamera- und 3 Objektbezogene Aktionen, die verschiedene Parameter erfordern. Grundparameter aller Aktionen ist ihre Dauer, weitere Parameter können Objekte, Bewegungen oder Trajektorienparameter sein.
Parameter

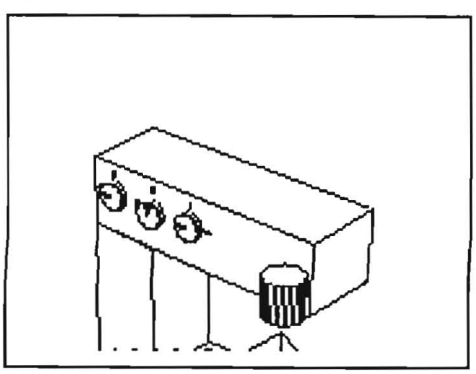
Alle Parameter beziehen sich auf Angaben, die von der konkreten geometrischen Realisierung der Objekte unabhängig sind. So wird beispielsweise nichts











in konkreten Brennweiten wie z.B. $50mm$ oder „Bildwinkel 30 Grad"angegeben, sondern stets in objektbezogenen Angaben: „Bildwinkel zur bildfüllenden Darstellung von Objekt X". Auch Bewegungen sind keine absoluten Angaben in (x, y, z)-Werten, sondern beziehen sich auf Bewegungsbeschreibungen, die im angeschlossenen Grafik-Design-System modelliert sind. Dies bildet die Voraussetzung für die Austauschbarkeit der Animationsrealisierung, die oben angesprochen wurde. Konkrete geometrische Daten werden nur vom Backend verarbeitet und auf der Planerseite in objektbezogene Angaben umgewandelt.

Umsetzung der elementaren Aktionen

Die Realisierung der elementaren Aktionen auf Scriptebene geschieht mithilfe von Schablonen (templates), d.h. zu jeder elementaren Aktion gibt es ein Gerüst aus elementaren Bewegungen in der Zwischensprache, das an den entsprechenden Stellen mit den jeweiligen Werten aufgefüllt wird. Das Ergebnis ist ein Script in der in Abschnitt 4.4 beschriebenen Zwischensprache, das die vollständige Beschreibung jedes Einzelbildes impliziert.

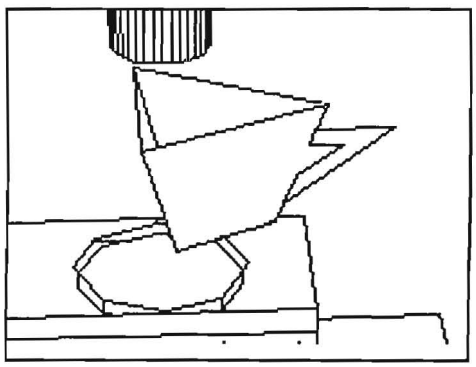
Die 7 Kameraaktionen

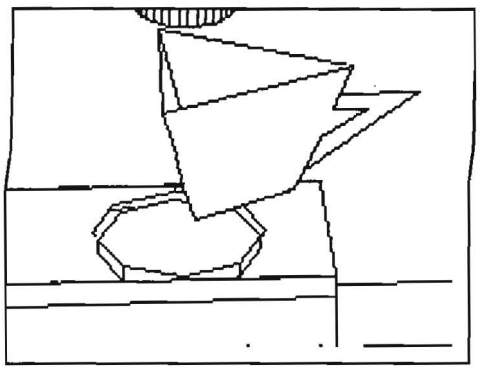
Die elementaren Kameraaktionen regeln die Bewegungen und Einstellungen der Kamera. Gesteuert werden drei Grundparameter, die Position der Kamera, der Kamerazielpunkt und die Brennweite des Objektivs (bzw. sein Bildwinkel). Die Menge der elementaren Kameraaktionen ist orthogonal in dem Sinne, daß es zu Kameraposition, Kamerazielpunkt und Brennweite je eine Variante zum Verändern und eine zum Belassen gibt (hierbei sei `X` ein Objekt und `t` eine Zeitangabe):

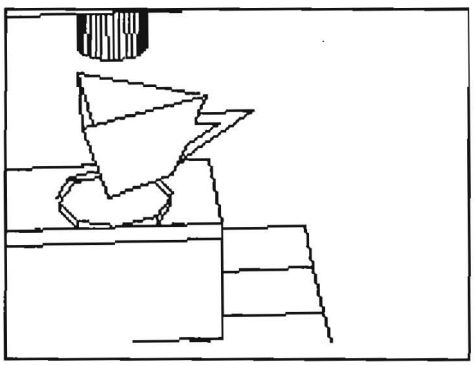
Kamera-	verändern	belassen
Pos.	`(move-cam-to-object X t)`	`(hold-cam-at-position t)`
Zielp.	`(move-aim-to-object X t)`	`(hold-aim-at-object X t)` `(hold-aim-at-position t)`
Bildw.	`(adjust-va-for-object X t)`	`(keep-view-angle t)`

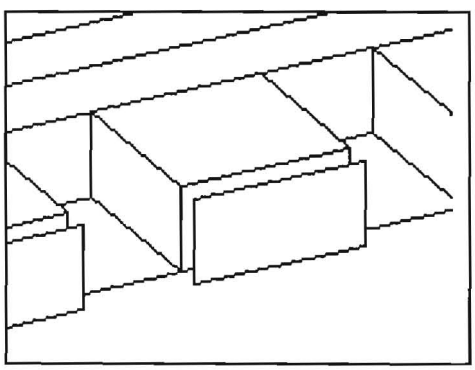
Das Bewegen der Kamera in die Position für Objekt `X` (`(move-cam-to-object X t)`) bedeutet nichts anderes, als daß die Kamera an eine Position bewegt wird, die die Komponente zur Perspektivenwahl aus TOPAS ([Sc94]) unter der Vorgabe berechnet hat, daß das Objekt `X` im Bild bei „normaler"Kameraeinstellung gut sichtbar ist, und zwar von seiner Vorderseite, sofern es eine solche besitzt.

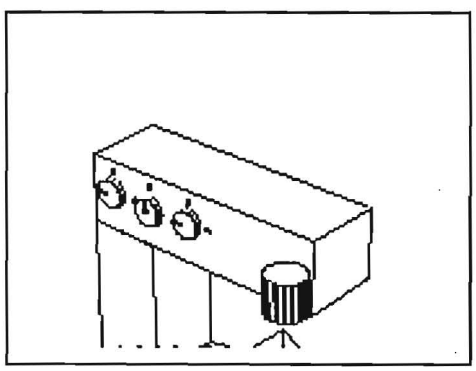
Die beiden Varianten beim Belassen des Kamerazielpunktes sind notwendig, da sich die Position der Objekte in der Zeit verändern kann. Somit braucht man eine Funktion, die den Zielpunkt objektbezogen fest läßt (d.h. auf dem Objekt











hält) und eine, die den Zielpunkt an seiner Position im Raum beläßt, unabhängig von Objektbewegungen. Bewegt sich das zugehörige Objekt nicht, so ist die Auswirkung der beiden Aktionen in der Animation identisch.

Die 3 objektbezogenen Aktionen

Die Bewegung von Objekten und die Explosion zusammengesetzter Objekte werden durch die folgenden elementaren Aktionen realisiert:

```
(move-object part motion start end time)
(move-object-and-aim part motion start end time)
(explode-object part time)
```

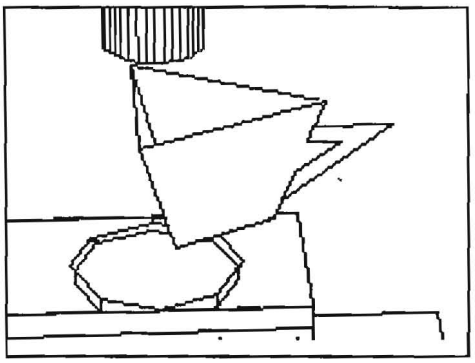
Zur Bewegung von Objekten gibt es also zwei verschiedene elementare Aktionen, die sich nur darin unterscheiden, daß bei der zweiten der Kamerazielpunkt mit der Objektbewegung mitgeführt wird. Die Notwendigkeit dieser zwei Varianten wird später bei der Erklärung der Kontextverwaltung und des Regelwerkes offensichtlich werden.

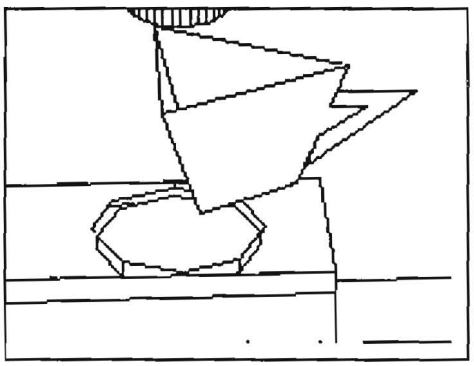
Die Beschreibung der Bewegungen erfolgt durch eine Datenstruktur, wie sie in Abschnitt 4.7 beschrieben ist. Die Parameter `start` und `end` bezeichnen verschiedene Stadien der Bewegung. Soll beispielsweise die erste Hälfte der Bewegung ausgeführt werden, so wird 0 als Startwert und 0.5 als Endwert angegeben, die zweite Hälfte erstreckt sich dann analog von Stadium 0.5 bis 1. Diese Möglichkeit zur Unterteilung einer Bewegung in verschiedene Phasen ist notwendig, um beispielsweise die Kameraführung auf die Bewegung abstimmen zu können (vgl. Kap. 4.3.3). Die Explosion zusammengesetzter Objekte wird vom angeschlossenen Grafik-Design-System berechnet (vgl. Kap. 4.7) und in Form von Achsenbewegungen ausgeführt.

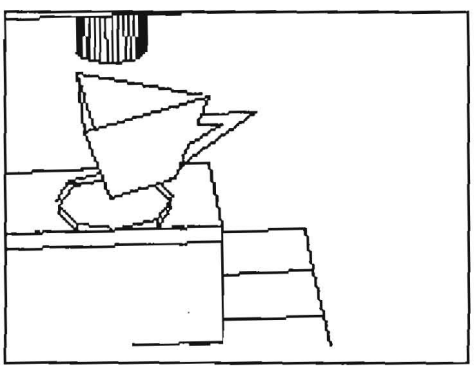
Mithilfe dieser insgesamt 10 elementaren Aktionen und ihrer zeitlich parallelen oder sequentiellen Kombination lassen sich beliebige Scripte aufbauen, die eine Baumstruktur besitzen. An den Blättern des Baumes stehen elementare Bewegungen, die Knoten werden jeweils durch eine Dekomposition gebildet.

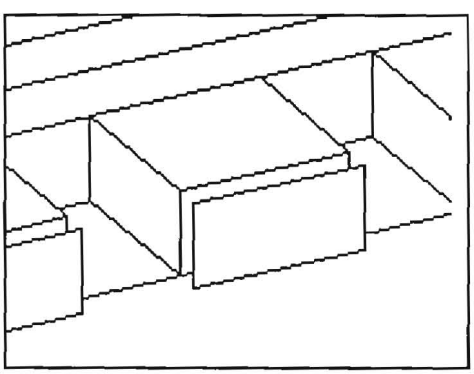
4.2.2 Unbedingte Dekomposition

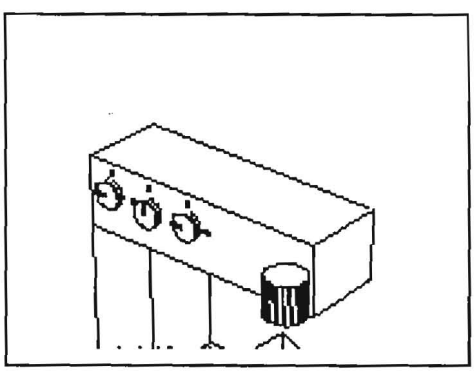
Zur Dekomposition eines Zieles in weitere Unterziele gibt es in BETTY zunächst zwei Varianten: die Zerlegung in zeitlich parallele Unterziele und die Zerlegung in zeitlich aufeinanderfolgende Unterziele. Sagt eine Regel beispielsweise aus, ein Ziel sei in mehrere zeitlich parallele Unterziele zu dekomponieren, so entsteht











dadurch eine Verzweigung der jeweiligen Sequenz in parallele Untersequenzen. Besagt die Regel, die Unterziele seien zeitlich nacheinander zu erfüllen, so entstehen aufeinanderfolgende Untersequenzen. Hierzu zwei Beispiele:

Eine Kamerafahrt, während der ein Objekt ständig im Bild bleibt, wird dadurch erreicht, daß zeitlich parallel die Kamera bewegt und der Kamerazielpunkt auf dem Objekt gehalten wird. Eine Kamerafahrt auf ein Objekt zu enthält in der Regel zu Ende eine kurze Ruhephase, damit die neue Position als statisch erkannt wird, das heißt, eine solche Kamerafahrt zerfällt zeitlich aufeinanderfolgend in die Fahrt selbst und eine Standbildeinstellung.

Solche statischen Regeln zur Dekomposition können jedoch nicht in allen Fällen angegeben werden. Oft hängt die Zergliederung eines Zieles von der momentanen Situation, vom Kontext ab und somit indirekt vom gesamten bisher geplanten Script.

4.2.3 Kontextverwaltung

Steht die Kamera beispielsweise in einer Position, in der ein bestimmtes Teil sichtbar ist, und sie soll laut Plan dieses Teil lokalisieren, so braucht keine Kamerafahrt geplant zu werden, sondern die Lokalisierung durch ein Zoom auf das Objekt kann direkt stattfinden. Steht die Kamera jedoch an einer anderen Position, aus der das Objekt nicht sichtbar ist, so ist eine Kamerafahrt oder ein Schnitt unumgänglich.

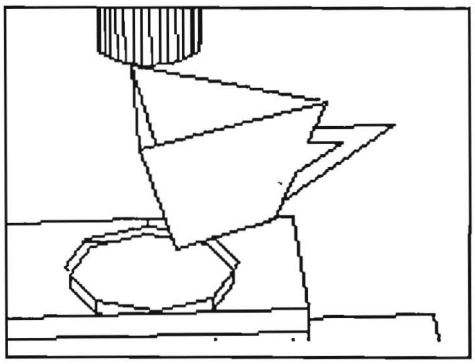
Das System braucht also ein bestimmtes Wissen über seinen aktuellen Zustand. Dieses Wissen muß außerdem wie schon weiter oben gefordert objektbezogene Aussagen liefern, wie z.B. „Die Kamera ist auf Objekt X gerichtet", nicht etwa „die Kamera ist auf Punkt (x, y, z) gerichtet". Außerdem muß der gespeicherte Zustand während der Planung immer auf dem aktuellen Stand gehalten werden.

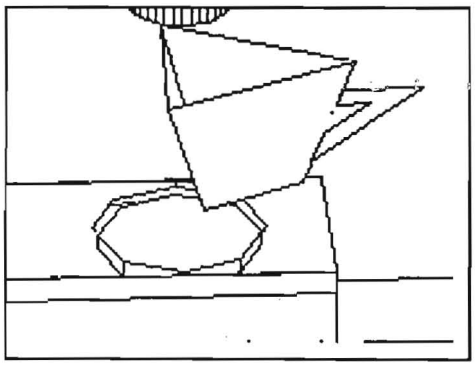
Eine solche Wissensbasis wird von der Planungskomponente verwaltet: Jede elementare Aktion, die geplant wird, trägt je nach ihren Seiteneffekten neue Information ein, löscht alte oder überschreibt sie, entsprechend den add- und delete- Listen in STRIPS.

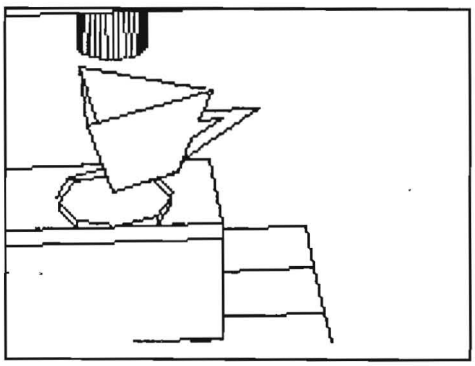
Ein Beispiel: Die elementare Aktion

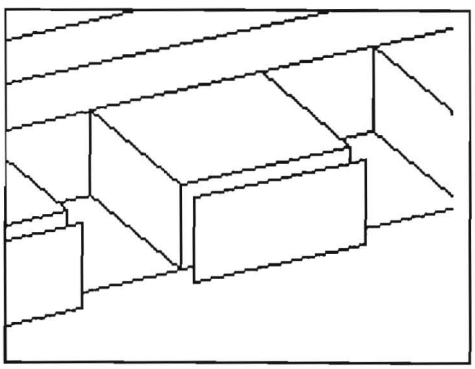
```
(move-aim-to-object X t)
```

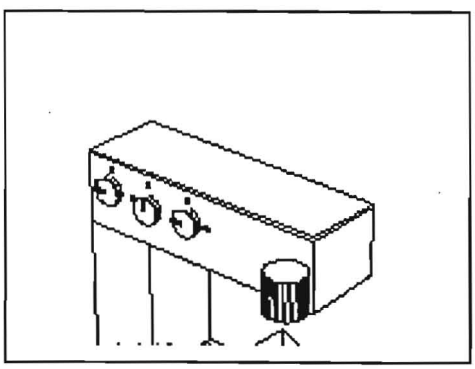
erzeugt den Eintrag, daß der Kamerazielpunkt jetzt auf Objekt `X` steht. Existierte vorher ein Eintrag, daß er auf Objekt `Y` gerichtet war, so wird dieser überschrieben











und durch obigen ersetzt. Die Aktion

```
(move-object X m s e t)
```

löscht diesen Eintrag wieder, da sich das Objekt möglicherweise aus dem Bildmittelpunkt wegbewegt hat. Die Aktion

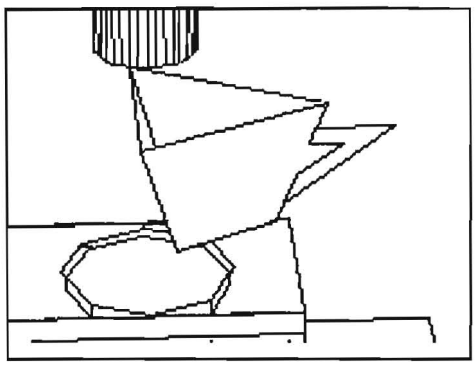
```
(move-object-and-aim X m s e t)
```

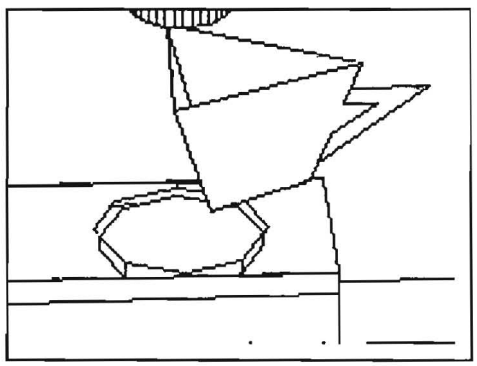
hingegen löscht den Eintrag nicht, da der Zielpunkt sich ja mitbewegt. Genau dieser Unterschied ist auch der Grund für die Existenz der elementaren Aktion `move-object-and-aim`, die ansonsten leicht durch eine parallele Dekomposition in `move-object` und `hold-aim-at-object` zu umschreiben wäre. Dabei ginge jedoch die Information verloren, daß die Kamera immer noch auf das Objekt gerichtet ist.

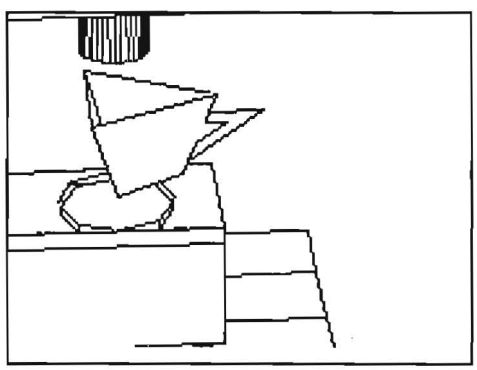
Die Kamerapositionen, die ein Objekt am besten zeigen, werden vom angeschlossenen Grafik-Designer (siehe 4.7) ausgewählt und der Kontexteintrag, daß die Kamera in Position für Objekt `X` ist, bedeutet, daß sie sich an der dort berechneten Position befindet.

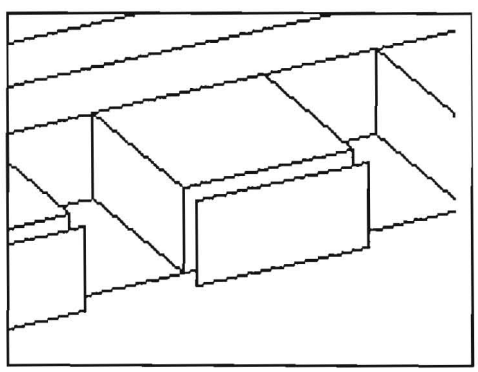
Aktion	trägt ein	löscht
`(move-cam-to-object X t)`	Kam. in Pos. für X	Kam. in Pos. für Y
`(hold-cam-at-position t)`	-	-
`(move-aim-to-object X t)`	Zielpunkt auf X	Zielpunkt auf Y
`(hold-aim-at-object X t)`	-	-
`(hold-aim-at-position t)`	-	-
`(adjust-va-for-object X t)`	Bildwinkel für X	Bildwinkel für Y
`(keep-view-angle t)`	-	-
`(move-object X m p q t)`	X in q bzgl. m	Kam. in Pos. für X Bildwinkel für X Zielpunkt auf X X in p bzgl. m
`(move-object-and-aim X m p q t)`	X in q bzgl. m	Bildwinkel für X X in p bzgl. m
`(explode-object X t)`	-	Bildwinkel für X Zielpunkt auf X

Genauso bedeutet der Kontexteintrag, daß der Bildwinkel für Objekt `X` eingestellt ist, daß bei der momentanen Kameraposition Objekt `X` bildfüllend gezeigt wird. Im einzelnen erzeugen die elementaren Aktionen jeweils die in der obigen









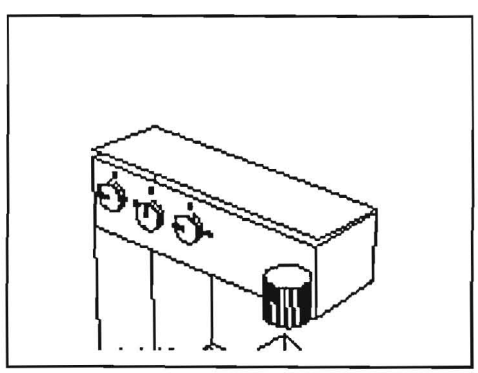


Tabelle genannten Kontextveränderungen, wobei X und Y verschiedene Objekte sind, t eine Zeitangabe und m eine Bewegungsbeschreibung ist.

Die Parameter p und q bezeichnen verschiedene Stadien der Bewegung m. Soll beispielsweise das mittlere Drittel der Bewegung m ausgeführt werden, so ist dies der Teil zwischen Stadium 0.3 und Stadium 0.7, die gesamte Bewegung erstreckt sich von Stadium 0 bis 1.

Werden verschiedene elementare Aktionen zeitlich aufeinanderfolgend geplant, so überschreiben die Kontextveränderungen der späteren Aktionen diejenigen der früheren Aktionen. Werden mehrere Aktionen geplant, die in der Animation zeitlich parallel ablaufen sollen, so wird die Priorität der Kontexteinträge durch die Reihenfolge der elementaren Aktionen in der betreffenden Dekompositionsregel bestimmt. Die zuunterst genannte und daher zuletzt geplante Aktion verändert auch als letzte den Kontext und ihre Veränderungen haben somit die höchste Priorität.

Durch diese Kontextverwaltung ist es dem Planer nun möglich, bedingte Dekompositionsregeln abzuarbeiten, deren Bedingungen sich auf den Kontext beziehen.

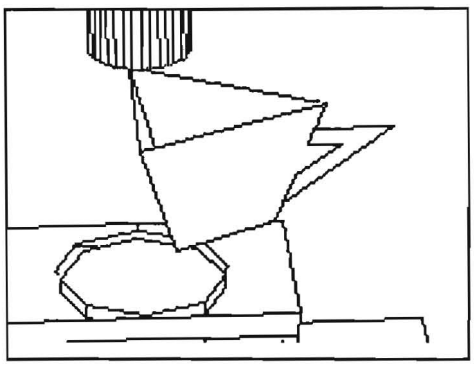
4.2.4 Bedingte Dekomposition

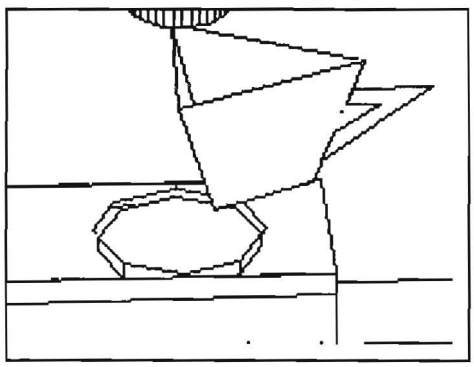
Soll beispielsweise ein Teil lokalisiert werden, so wird in der zugehörigen Regel abgefragt, ob das Objekt überhaupt schon im Bild ist. Ist dies nicht der Fall, so wird je nach aktueller Kameraposition ein Schnitt oder eine Fahrt in eine geeignete Position geplant. Solche Anfragen können logisch beliebig miteinander verknüpft werden und sich auf Objekte, deren Oberobjekte, Bewegungszustände oder die Kamera beziehen. So sind komplexe Bedingungen realisierbar, wie z.B.

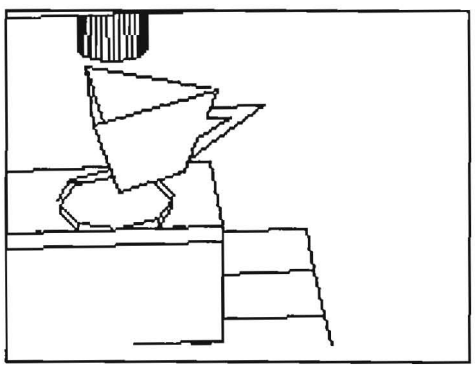
> „Falls die Kamera auf das Objekt X gerichtet ist, dessen Oberobjekt komplett im Bild sichtbar ist und sich Teil X in seiner Ausgangsposition befindet"

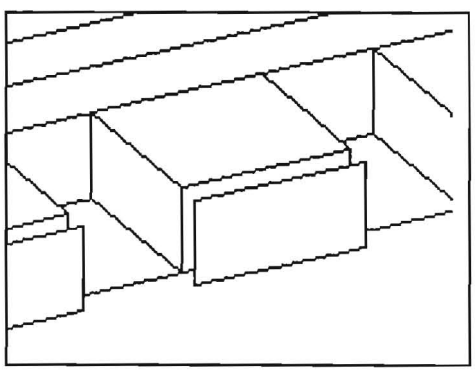
Aufgrund solcher Bedingungen kann dann zu verschiedenen Unterzielen verzweigt werden, die jeweils wieder unterschiedliche Dekompositionen implizieren. Um die Auswirkungen einer bedingten Dekomposition an einem konkreten Beispiel klar zu machen, wollen wir folgendes Visualisierungsziel betrachten, das sich auf das Bewegen eines Knopfes an einem Gerät bezieht:

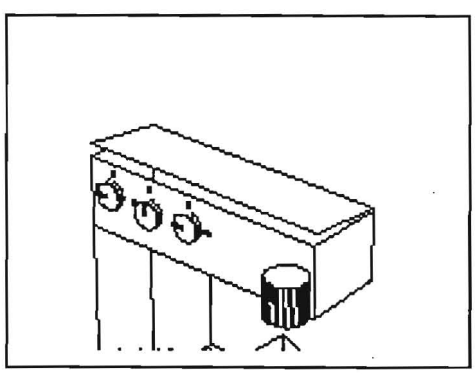
```
(show-object-motion 'rechts-drehen' 'Knopf-3' 4)
```











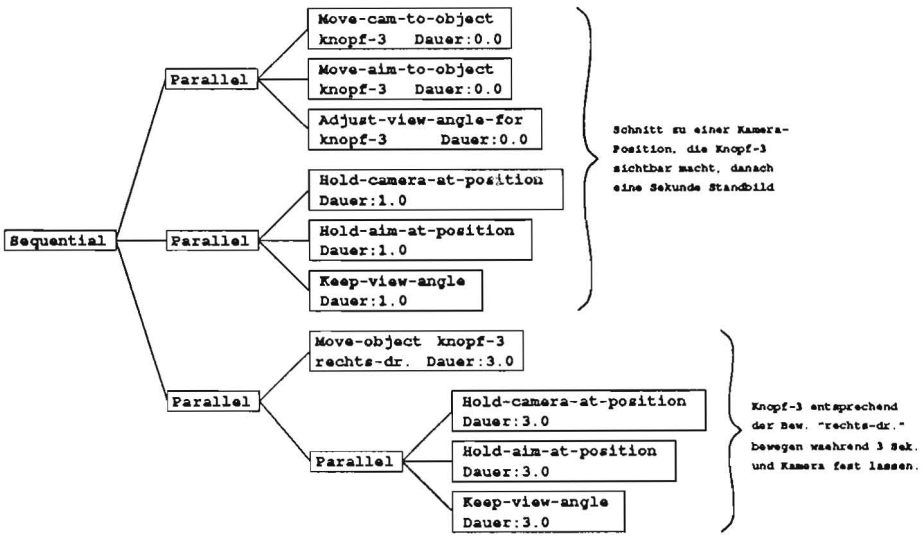
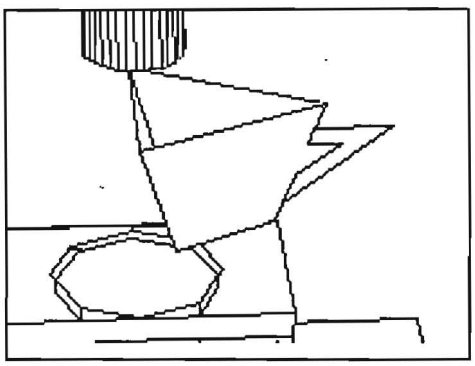


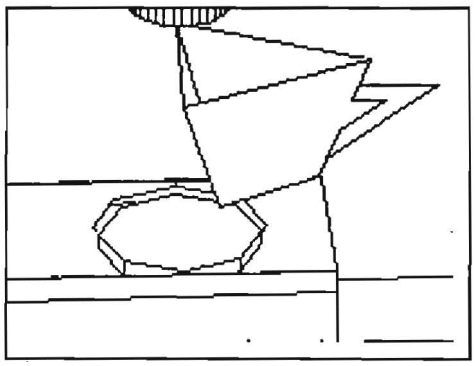
Abbildung 12: Bedingte Dekomposition – 1. Variante

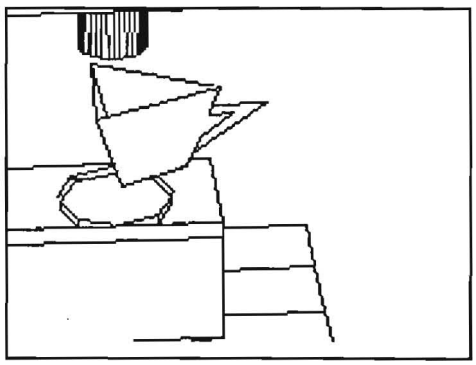
Die Dekomposition erfolgt durch die bedingte Dekompositionsregel für das Visualisierungsziel `show-object-motion`, die in Anhang A mit all ihren Verzweigungen aufgelistet ist. Im ersten Fall (Bild 12) ist die Kamera zu Ende der vorhergehenden Einstellung auf der falschen Seite der Szenerie. So muß ein Schnitt (d.h. eine Kamerabewegung in Zeit 0) zu einer geeigneten Kameraposition erfolgen, bevor der Knopf gezeigt werden kann.

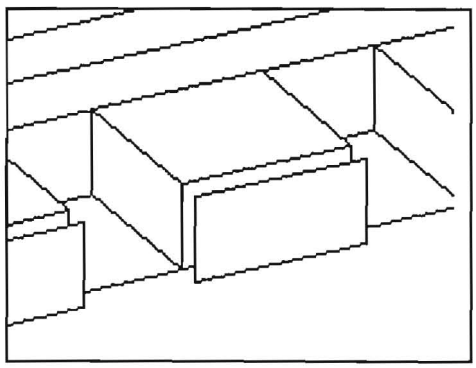
Im zweiten Fall (Bild 13) befindet sich die Kamera bereits in einer für Knopf-3 geeigneten Position, da dieser in einer vorangehenden Einstellung bereits lokalisiert oder eine andere Bewegung damit gezeigt wurde. In der Kontext-Datenbasis ist diese Kameraposition vermerkt.

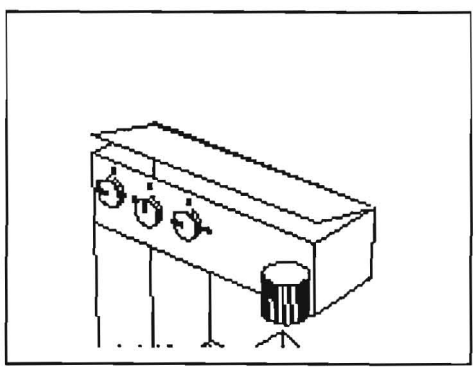
Wie anhand der Bilder ersichtlich ist, entstehen zwei unterschiedliche Dekompositionsbäume und folglich zwei völlig unterschiedliche Animationsteile, je nach dem Kontext, in dem die Szene geplant wird. Anschaulich werden die drei verschiedenen Dekompositionsarten und ihre Auswirkungen im Script nochmals in Bild 14 dargestellt.











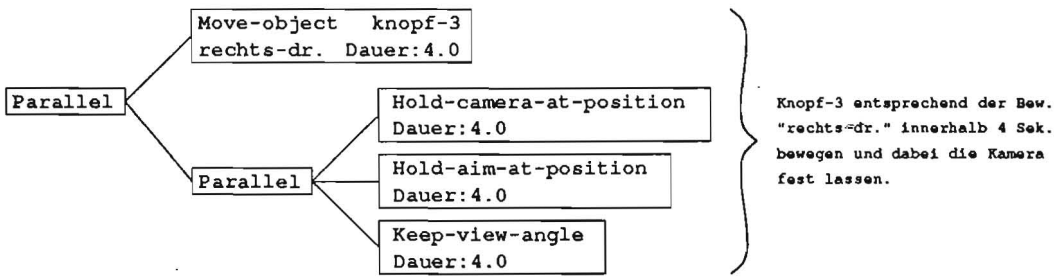


Abbildung 13: Bedingte Dekomposition – 2. Variante

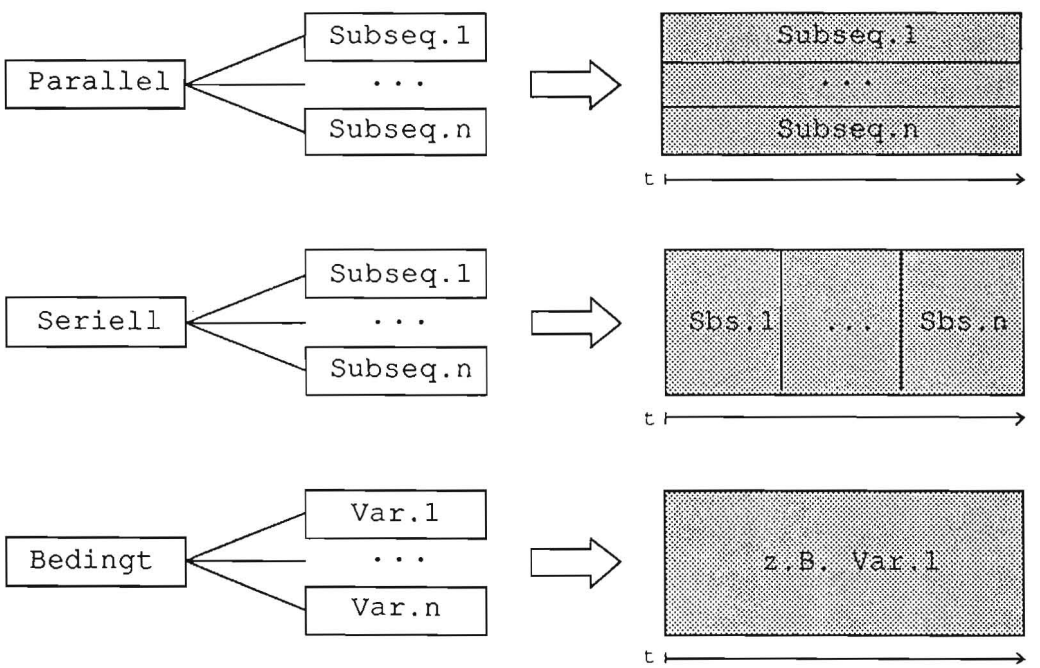
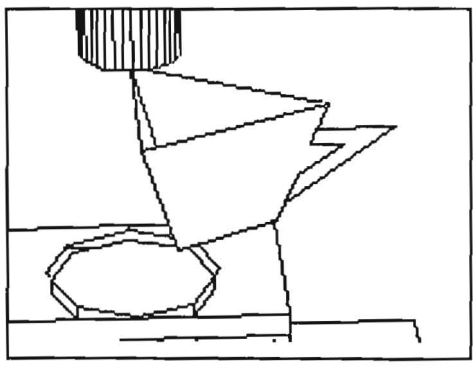
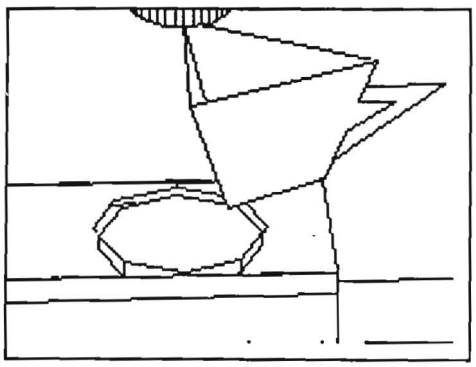
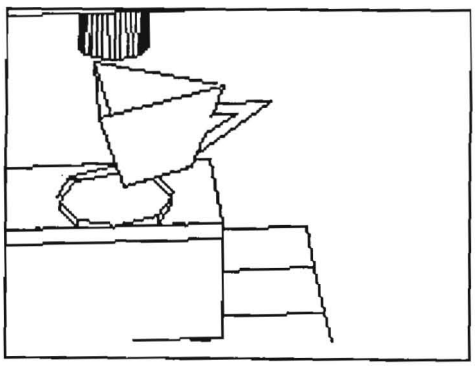
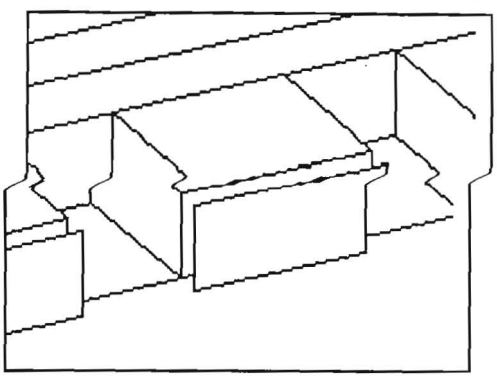


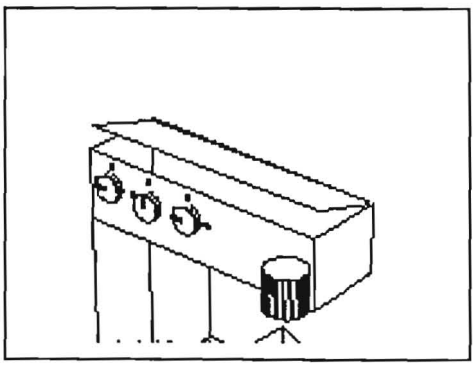
Abbildung 14: Die drei Arten der Dekomposition und ihre Auswirkungen im Script











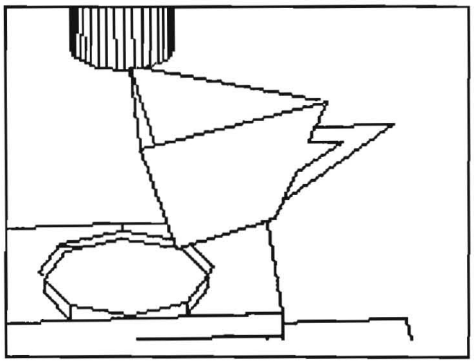
4.2.5 Der Planungsalgorithmus

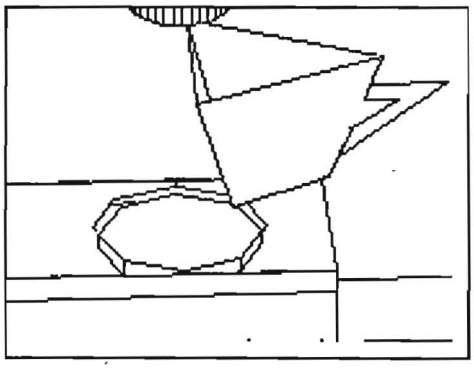
Nachdem nun die grundlegenden Begriffe eingeführt sind, läßt sich der Algorithmus zur vollständigen Dekomposition eines Visualisierungsziels im einzelnen wie folgt beschreiben:

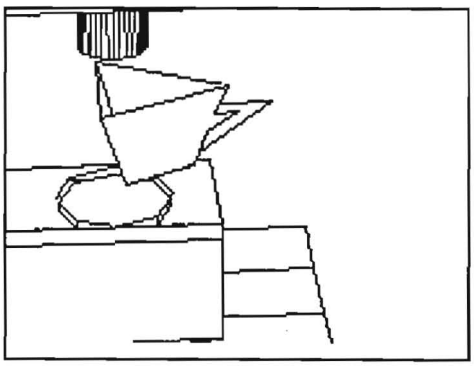
Dekomposition (Z) :=
  Falls Z elementare Aktion ist
    Verändere den Kontext entsprechend den
    Seiteneffekten der Aktion Z
    Ergebnis := Übersetzung von Z mittels Template
  Andernfalls
    Finde Dekompositionsregel R zu Z
    Falls R eine bedingte Regel ist
      mit Bedingungen B_i und Unterzielen Z_i
      Falls erste erfüllte Bedingung B_k
        Ergebnis := Dekomposition (Z_k)
      Falls kein B_k erfüllt
        Ergebnis := NIL
    Falls R eine unbedingte Regel ist mit Unterzielen Z_1 bis Z_n
      in Anordnung O (Parallel oder Sequentiell)
      Für alle Unterziele $Z_i, i = 1..n$ berechne
        D_i := Dekomposition (Z_i)
      Je nach Anordnung O setze
        Ergebnis := (:PARALLEL D_1, .. ,D_n)
      oder
        Ergebnis := (:SEQUENTIAL D_1, .. ,D_n)
    Falls keine Regel existiert: Fehler

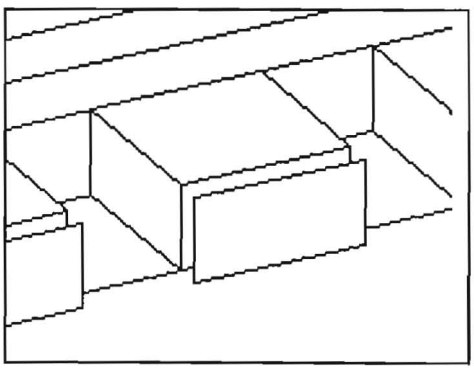
Bei der Zergliederung eines Visualisierungszieles in einen Scriptbaum wird jeder Ast des Baumes zuerst bis zu seinem Ende expandiert (Tiefensuche). Dies hat den Vorteil, daß zu jedem Zeitpunkt die schon geplanten elementaren Aktionen alle im Detail bekannt sind und ihre Auswirkungen auf den aktuellen Kontext an dieser Stelle direkt mit berücksichtigt werden können. Nach einer Übersetzung der elementaren Aktionen in elementare Scriptteile bilden diese im Zusammenspiel das hierarchisch aufgebaute Script.

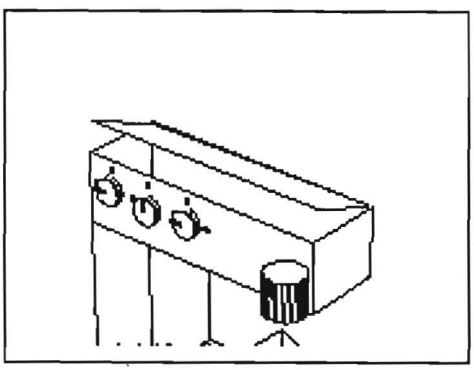
Die wirkliche Arbeitsweise des Planers ist im Detail noch etwas komplexer, da auf jeder Ebene der Dekomposition die Parameter des aktuellen Zieles und seiner Unterziele verwaltet werden müssen, die möglicherweise Terme enthalten











und weitere Berechnungen erfordern. Auch die Übersetzung der elementaren Aktionen selbst ist nicht so trivial, wie es den Anschein haben mag, da hierbei die Kontextveränderung nicht nur symbolisch, sondern auch in der Modellwelt des Grafiksystems erfolgt, damit diese beiden Repräsentationen zu jedem Zeitpunkt übereinstimmen. Weitere Details sind den in Anhang B abgedruckten Auszügen aus dem LISP- Quellcode zu entnehmen.

4.3 Das Regelwerk BETTYs

Das Regelwerk, das in Anhang A komplett aufgelistet ist, umfaßt derzeit 49 Dekompositionsregeln und spiegelt die verschiedenen Beschreibungsebenen einer Animation, wie sie in Kapitel 2.4 angegeben ist, wieder. Aus elementaren Aktionen werden zunächst Grundbausteine zur Kameraführung und zur Objektbewegung aufgebaut, aus denen dann komplexere Funktionen zusammengesetzt werden können.

Die Syntax und Semantik der Dekompositionsregeln sowie die gestalterischen und filmtechnischen Überlegungen, die dem bestehenden Regelwerk zugrunde liegen, wollen wir uns nun im Einzelnen anschauen.

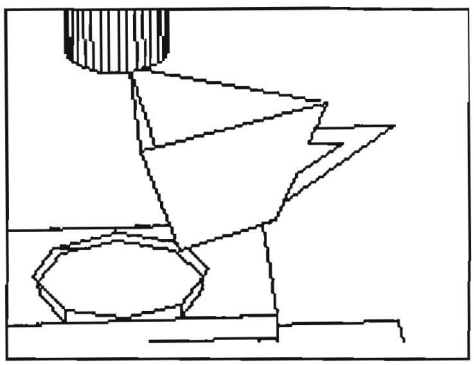
4.3.1 Syntax der Dekompositionsregeln

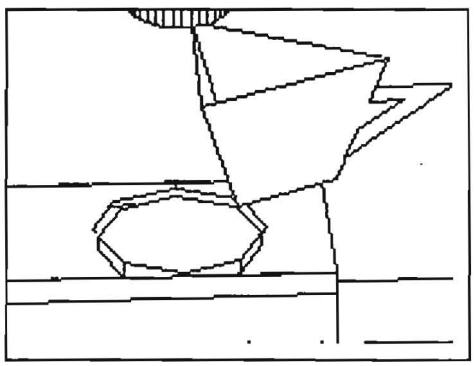
Die Syntax der kontextunabhängigen Dekompositionsregeln läßt sich formal wie folgt beschreiben:

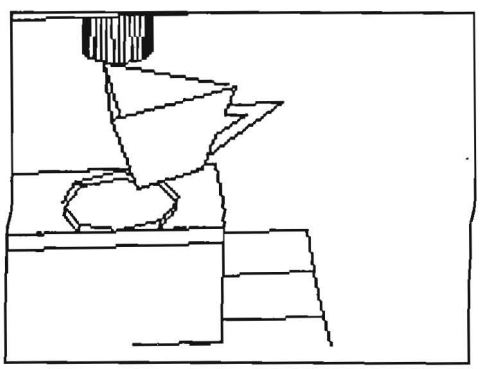
(anim-rule Z (arg_1 .. arg_n)
:PARALLEL | :SEQUENTIAL
($Unterziel_1$ $args...$)
...
($Unterziel_n$ $args...$))

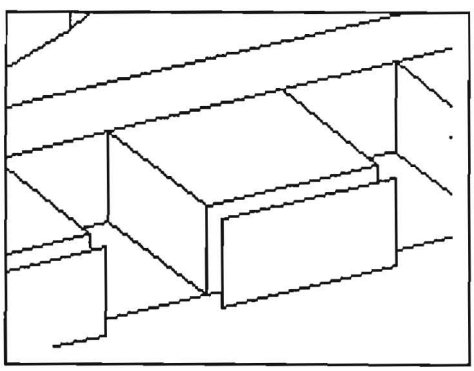
Hierbei ist Z der Name der Dekompositionsregel bzw. des Ziels, dessen Dekomposition die Regel angibt, da zu jedem Ziel jeweils nur eine Dekompositionsregel existiert. Anstelle einer Konfliktauflösung bei mehreren anwendbaren Dekompositionsregeln tritt wie oben beschrieben die Auswahl zwischen verschiedenen Alternativen in einer bedingten Dekomposition.

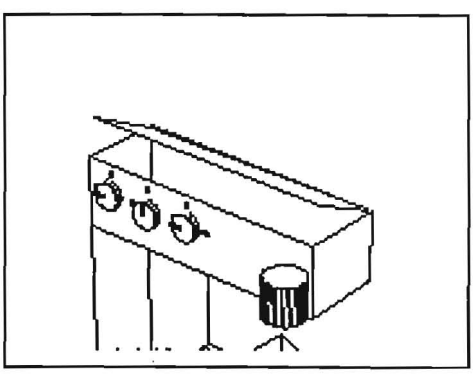
Die Argumente arg_1 .. arg_n sind die Parameter des Zieles Z. Die Parameter können in der Formulierung der Unterziele direkt oder in beliebig geschachtelten Termen wieder verwandt werden. Ist der Parameter `time` beispielsweise angegeben, so kann in den Unterzielen damit gerechnet werden, z.B. `(* time 0.5)`











oder (`min time 1`). Die zur Verfügung stehenden Operationen für numerische Argumente sind dabei $*, /, +, -, min, max$ im jeweiligen üblichen mathematischen Sinne. Für Parameter, die Objekte bezeichnen, steht außerdem die Funktion `superobject` zur Verfügung, die das nächsthöhere Objekt in der Teil-von-Hierarchie liefert.

Die Syntax der kontextabhängigen Regeln beinhaltet stattdessen für jedes mögliche Unterziel eine Anwendbarkeitsbedingung. Formal läßt sie sich so beschreiben:

(`anim-rule` Z (arg_1 .. arg_n)
:COND
($Bedingung_1$ ($Unterziel_1$ $args...$))
...
($Bedingung_n$ ($Unterziel_n$ $args...$)))

wobei die Unterziele entweder elementare Aktionen sind oder wieder durch Regeln aufgelöst werden. Das Zeichen `T` ist die leere Bedingung, die immer zutrifft. Sie taucht gewöhnlich im letzten Glied einer kontextabhängigen Regel auf und bewirkt dort, daß keine leere Untersequenz geplant wird, sondern als Default dieses letzte Unterziel eingesetzt wird. Die Syntax der Bedingungen lautet:

(`and` | `or` | `not` (obj_1 $oper_1$ val_1)
...
(obj_n $oper_n$ val_n))

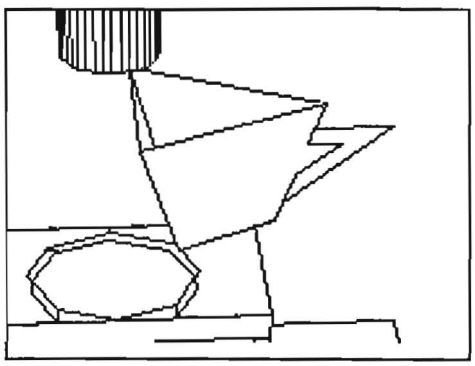
wobei $obj_i (1 \leq i \leq n)$ ein Objekt bezeichnet (insbesondere z.B. die Kamera), $oper_i$ eine Operation, die mit diesem Objekt auszuführen ist (bei der Kamera beispielsweise Position, Zielpunkt und Brennweite) und val_i den dazugehörigen Wert oder Zustand. Um die Formulierung und Anwendung der Bedingungen deutlich zu machen, hierzu ein Beispiel: Die Abfrage

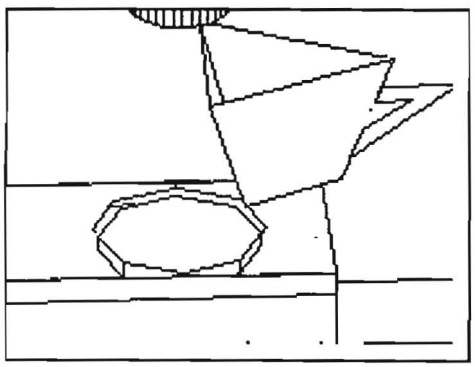
> „Ist die Kamera in Position für Objekt Knopf-1 und Knopf-1 bildfüllend?"

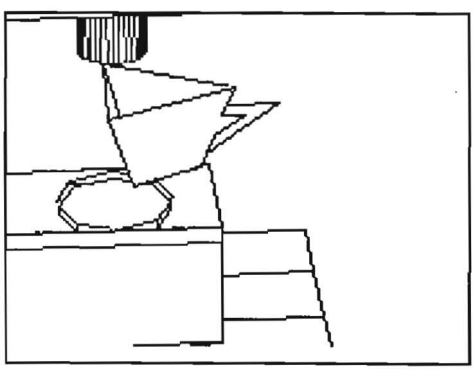
lautet in dieser Notation

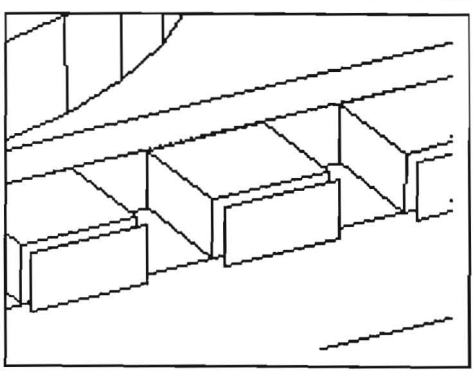
```
(and (camera cam-pos knopf-1)
     (camera view-angle-for knopf-1))
```

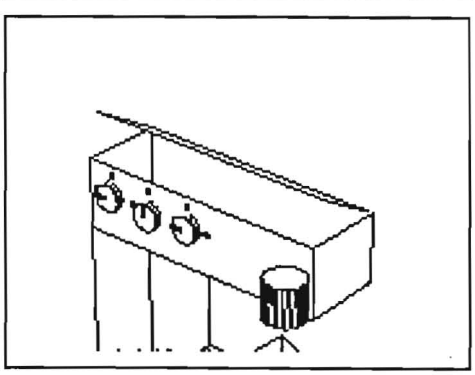
An dieser Stelle und während der folgenden beiden Abschnitte empfiehlt es sich, einen Blick in Anhang A zu werfen, der in Form der bestehenden Regeln umfangreiches Anschauungsmaterial bietet. Die Bedeutung der Regelkonstrukte wird dabei intuitiv wohl klar, trotzdem möchte ich sie nochmals erläutern.











4.3.2 Semantik der Dekompositionsregeln

Zunächst wollen wir die Semantik der unbedingten Dekompositionsregeln betrachten: Eine Regel der Form

```
(anim-rule Z (arg1 .. argn)
   :PARALLEL
   (Unterziel1 args...)
   ...
   (Unterzieln args...))
```

schreibt vor, daß das Unterziel Z durch die zeitlich parallele Erledigung der Unterziele $Unterziel_1$ bis $Unterziel_n$ erreicht wird. Auf Ebene des Scripts bedeutet das, daß die Sequenz Z in n parallele Untersequenzen zerfällt.

Die Parameter des Zieles Z, die im Regelkopf als Stellungsparameter angegeben werden, können innerhalb des Regelkörpers direkt oder innerhalb von beliebig tief geschachtelten Termen verwandt werden. Terme sind im üblichen mathematischen Sinne als vollständig geklammerte Terme definiert, in denen die binären Präfix-Operationen

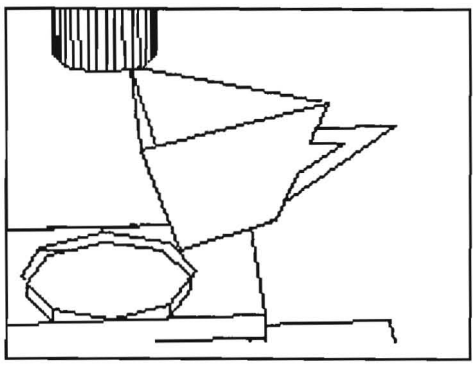
*,/,+,-,min,max

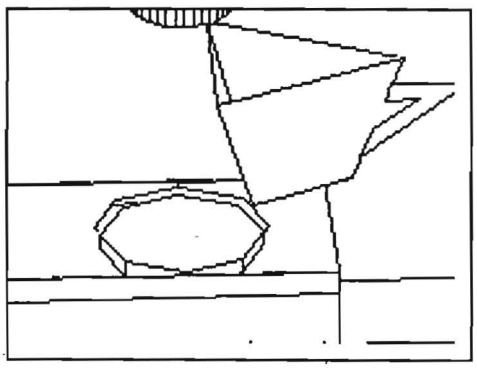
in ihrer jeweiligen üblichen mathematischen Bedeutung erlaubt sind. Die Notation entspricht derjenigen für Terme in LISP. Für Parameter, die Objekte bezeichnen, liefert die unäre Operation `superobject` das nächsthöhere Objekt in der Teil-von-Hierarchie.

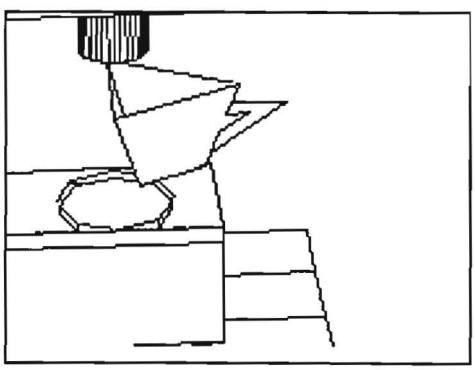
Genau analog mit nur zwei Zusätzen ist die Semantik der sequentiellen Dekompositionsregeln anzugeben: Eine Regel der Form

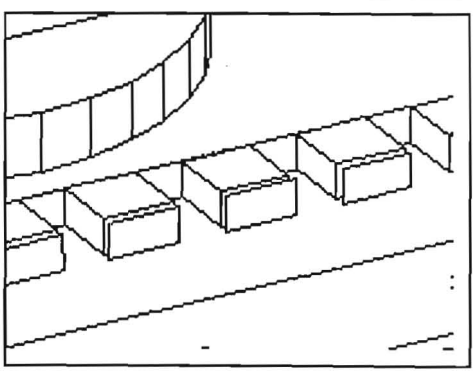
```
(anim-rule Z (arg1 .. argn)
   :SEQUENTIAL
   (Unterziel1 args...)
   ...
   (Unterzieln args...))
```

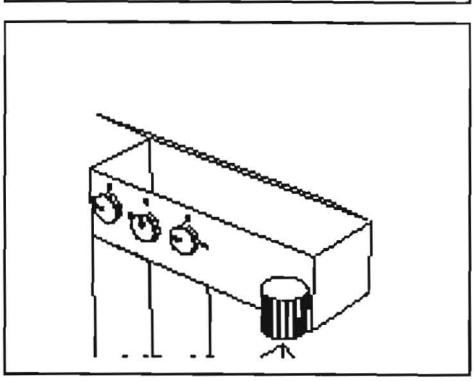
schreibt vor, daß das Unterziel Z durch die zeitlich aufeinanderfolgende Erledigung der Unterziele $Unterziel_1$ bis $Unterziel_n$ erreicht wird. Auf Ebene des Scripts bedeutet das, daß die Sequenz Z in n aufeinanderfolgende (sequenzielle) Untersequenzen zerfällt. Für die Parameter gilt das oben gesagte. Die Reihenfolge, in der die Unterziele hingeschrieben werden, bestimmt die Reihenfolge, in der sie zu erfüllen sind (bzw. auf Ebene des Scripts die Reihenfolge der Untersequenzen).











Randbedingung: Die zur Verfügung stehende Zeit (Parameter `time`) muß durch die Verwendung von Termen so auf die Unterziele verteilt werden, daß in der Summe die gesamte zur Verfügung stehende Zeit aufgebraucht wird.

Die Semantik der bedingten Dekompositionsregeln lautet wie folgt: Eine Regel der Form

(anim-rule Z (arg_1 .. arg_n)
:COND
($Bedingung_1$ ($Unterziel_1$ $args...$))
...
($Bedingung_n$ ($Unterziel_n$ $args...$)))

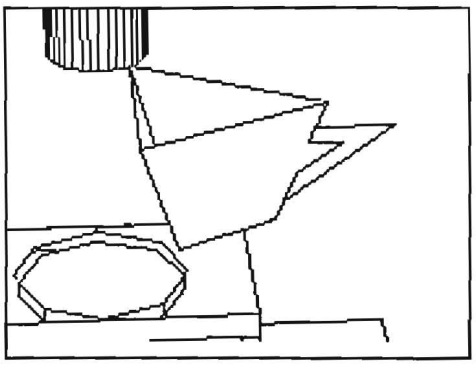
schreibt vor, daß das Unterziel Z durch die Erledigung eines der Unterziele $Unterziel_1$ bis $Unterziel_n$ erreicht wird. Auf Ebene des Scripts bedeutet das, daß die Sequenz Z höchstens eine Untersequenz hat, und zwar diejenige, deren Bedingung erfüllt war. Sind die Bedingungen mehrerer Unterziele erfüllt, so wird davon nur das erste genommen, ist keine der gestellten Bedingungen erfüllt, so ergibt sich die leere Sequenz. Das Zeichen `T` ist die leere Bedingung, die immer zutrifft. Sie taucht gewöhnlich im letzten Glied einer kontextabhängigen Regel auf und bewirkt dort, daß keine leere Untersequenz geplant wird, sondern gewissermaßen als Default dieses letzte Unterziel eingesetzt wird. Für die Parameter gilt wieder das oben gesagte.

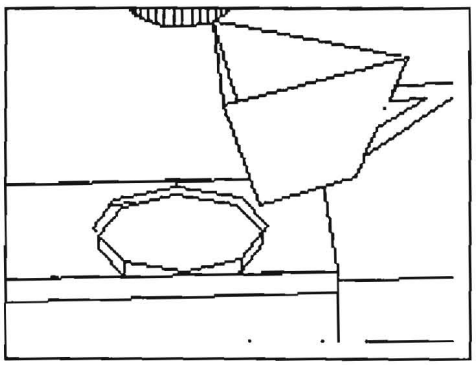
4.3.3 Aufbau des Regelwerks

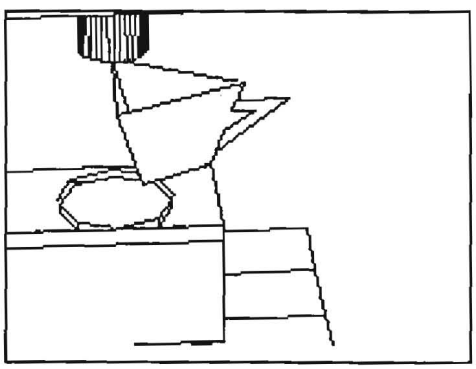
Gliederung in verschiedene Ebenen

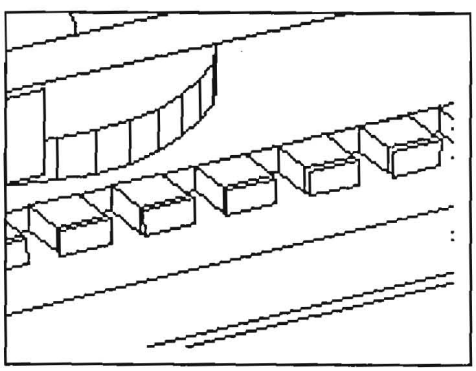
Das Regelwerk spiegelt, wie eingangs erwähnt, die verschiedenen Beschreibungsebenen einer Animation, wie sie in Kapitel 2.4 angegeben sind, indirekt wieder. Aus elementaren Aktionen werden zunächst Grundbausteine zur Kameraführung und zur Objektbewegung aufgebaut, aus denen dann komplexere Funktionen zusammengesetzt werden können.

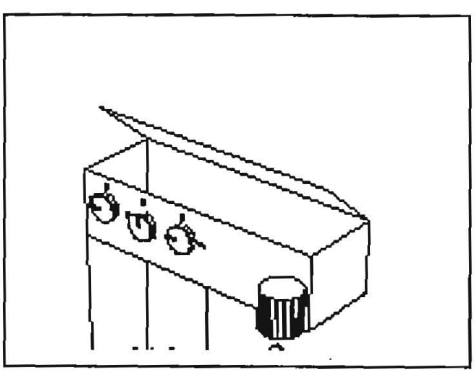
Nun entspricht allerdings nicht zwangsläufig dem Übergang von einer Abstraktionsebene zur nächsten eine einzelne Dekomposition. Eine Ebene kann mehrere Dekompositionsschritte beinhalten oder umgekehrt ein einzelner Schritt zwei Ebenen überspringen. Die Analogie zwischen dem Aufbau des Regelwerkes und











der in Abschnitt 2.4 eingeführten Beschreibungshierarchie besteht deshalb mehr auf inhaltlicher als auf formaler Ebene.

Zugunsten einer überschaubaren Hierarchie des Regelwerks und der entstehenden Animationsscripte sowie einer verständlichen Terminologie wurde außerdem auf eine in allen Fällen optimale Zergliederung bzgl. der Größe des Scriptbaumes verzichtet. So kann es beispielsweise vorkommen, daß zwei parallele Untersequenzen wiederum nur parallele Untersequenzen enthalten, was eigentlich in einer einzigen parallelen Dekomposition auch erreichbar gewesen wäre. Die zusätzlichen Ebenen entsprechen jedoch immer auch sinngemäßen Aufteilungen, wie z.B. einer Gliederung in Kamera- und Objektbewegung.

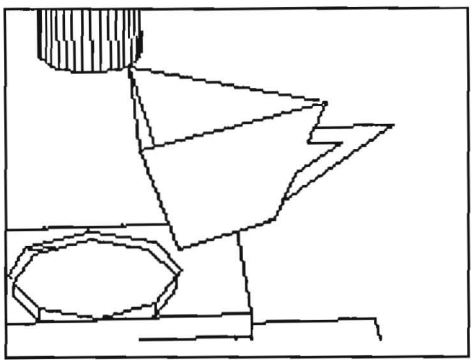
Diese Vorgehensweise erleichtert das Formulieren und Verfeinern Domänenunabhängiger Regeln enorm und ermöglicht es außerdem, den Gesamtüberblick zu behalten, da das fertige Script beim möglichen Nacheditieren alle einzelnen Dekompositionen wiederspiegelt und somit vom Standpunkt der Regelformulierung verständlich bleibt.

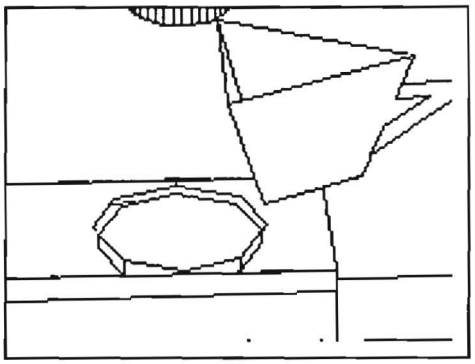
Rekursive Regeln

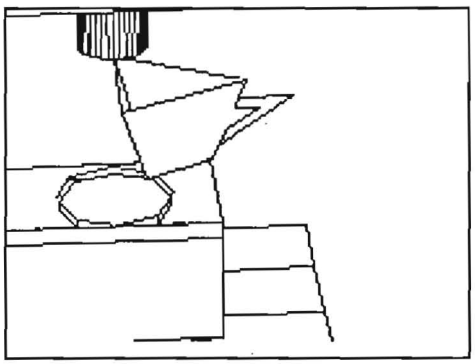
Prinzipiell besteht die Möglichkeit, die von Betty benutzten Dekompositionsregeln so zu formulieren, daß die Dekomposition eines Unterzieles in Abhängigkeit vom Kontext wiederum das gleiche Unterziel impliziert. Damit diese Rekursion jedoch endet, muß zwischen zwei Rekursionsschritten eine Kontextveränderung stattfinden. Ändert sich der Kontext nicht, so führt die Konstellation zu einer endlosen Rekursion.

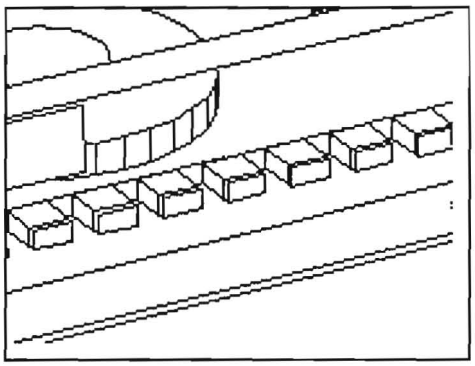
Solche Rekursionen können in einem einzigen Dekompositionsschritt auftreten, können sich aber auch über mehrere verschiedene Regeln erstrecken, und sind dann nicht mehr abzufangen. Um eine endlose Rekursion des Animationsplaners zu vermeiden, ist es daher empfehlenswert, rekursive Dekompositionsregeln nur in einfachen und leicht überschaubaren Fällen zu formulieren, in denen auch sichergestellt werden kann, daß entsprechende Kontextveränderungen stattfinden, die die Rekursion beenden.

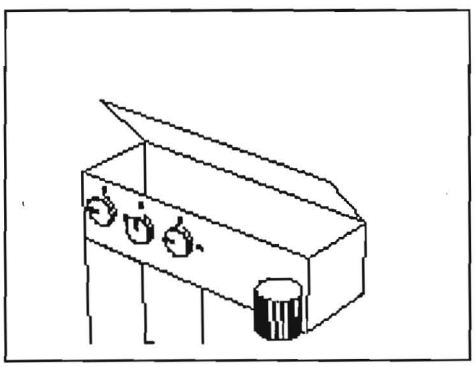
Ein einfaches Beispiel für eine unkritische Verwendung rekursiver Dekompositionsregeln wäre eine Rekursion über der Teil-von-Hierarchie der gezeigten Objekte, indem z.B. die Lokalisierung eines Objektes die vorherige Lokalisierung seines Oberobjektes impliziert. Da der Teil-von-Baum endlich und azyklisch ist, gibt es irgendwann einmal ein oberstes Oberobjekt, das als erstes lokalisiert wird womit die Rekursion beendet ist.











Codierung Filmtechnischer Verfahren

Die endgültige Realisierung der Visualisierungsziele berücksichtigt einige grundlegende Forderungen der Filmtechnik, wie sie auch in [Ka90] genannt werden. So wird beispielsweise die Bewegungsachse von Objekten mit der Kamera nicht gekreuzt, während eine Bewegung stattfindet (line crossing error), indem die betreffenden Regeln die Kameraposition für die Dauer der Bewegung konstant halten und die notwendige Verfolgung des Objektes alleine durch Schwenken und Zoomen der Kamera bewirken. Außerdem werden dem Betrachter (ebenfalls durch geeignete Formulierung der Regeln) immer wieder Ruhephasen eingeräumt, z.B. am Ende einer Kamerafahrt oder einer Objektbewegung, damit die neue Situation als statisch erkannt werden kann.

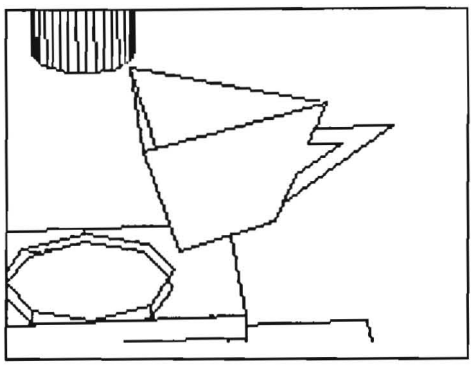
Die visuellen Gestaltgesetze, wie sie u.a. in [Ka26, Ma75] zur Sprache kommen, haben die Formulierung der Regeln ebenfalls beeinflußt, indem beispielsweise gleichartige Objekte immer ins Verhältnis zu ihrem Oberobjekt gesetzt werden, wenn sie nacheinander lokalisiert werden (Gesetz der Gleichheit). Ein Beispiel dafür ist in Daumenkino Nr. 4 zu sehen.

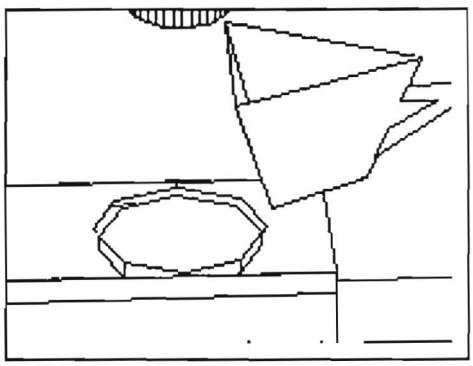
Zeigen von Objektbewegungen

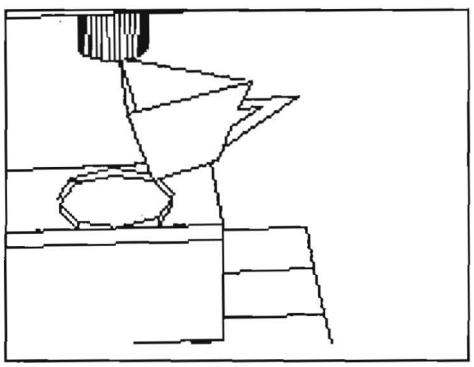
Ein besonderes Problem stellt die Verfolgung von Objektbewegungen mit der Kamera dar: Handelt es sich um Drehungen oder kleinräumige Bewegungen, so kann die Kameraeinstellung in der Regel unverändert bleiben. Werden Objekte jedoch über größere Strecken hinweg bewegt, so verlassen sie möglicherweise das Bildfeld, müssen also mit der Kamera verfolgt werden.

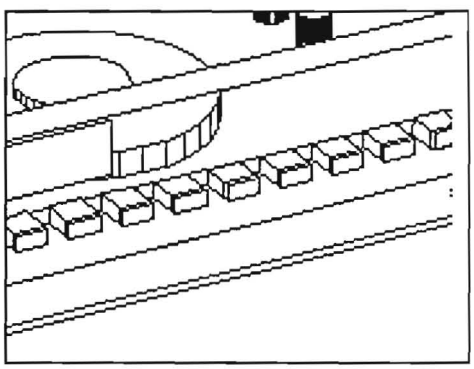
Hält man die Kamera nun statisch auf das Objekt, so entsteht der überraschende Eindruck, das Objekt verbleibe an einer festen Stelle im Raum und die gesamte Umgebung bewege sich fort (siehe Daumenkino 1). Hält man die Kamera hingegen fest in Relation zur Umgebung, so verschwindet möglicherweise das Objekt aus dem Bild (Daumenkino 2).

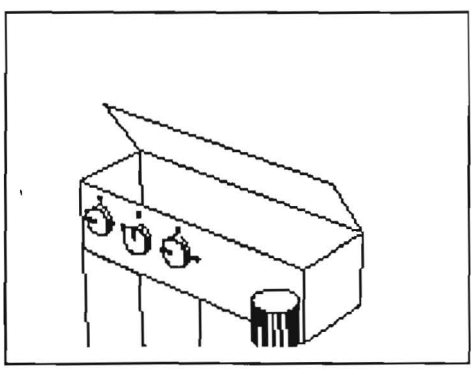
Gelöst wurde dieses Problem durch eine kombinierte Bewegung der Kamera aus drei Phasen (Daumenkino 3): Zu Beginn der Bewegung bleibt die Kamera fest am Ort und läßt die Objektbewegung im Bild sichtbar werden. In der zweiten Phase wird die Kamera dann so geschwenkt, daß sich das bewegte Objekt zu Ende der zweiten Phase im Bildmittelpunkt befindet. In der dritten Phase wird die Objektbewegung mitverfolgt wie in Beispiel 1. Durch die vor-









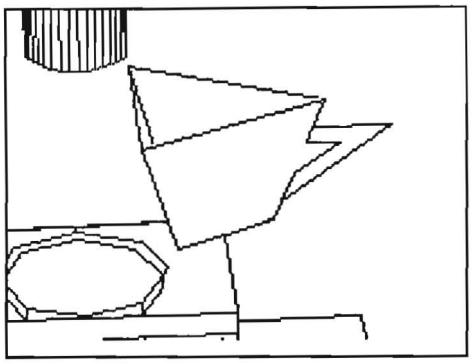


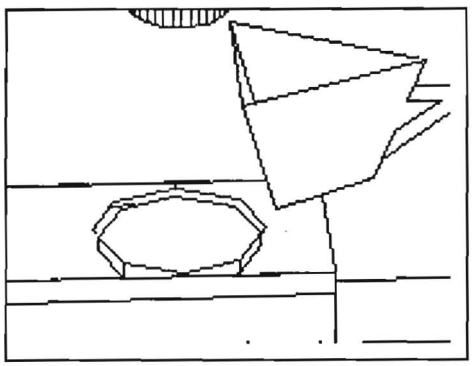
angegangenen Kamerabewegungen ist jetzt jedoch klar, daß sich das Objekt bewegt, nicht die Umgebung. Der Bildwinkel bleibt während aller drei Phasen erhalten, damit für den Betrachter der Bezug des Objektes zur Umgebung nicht verloren geht. Die Dekompositionsregel zur Realisierung dieses Verfahrens heißt `show-object-motion` und findet sich in Anhang A.

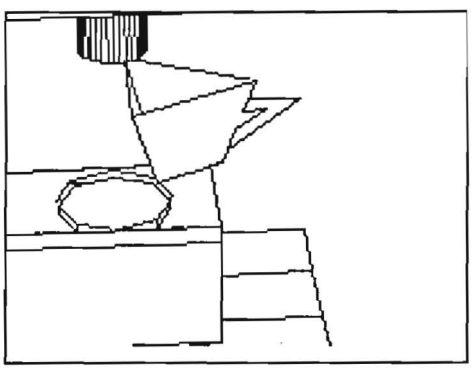
Lokalisation von Teilen

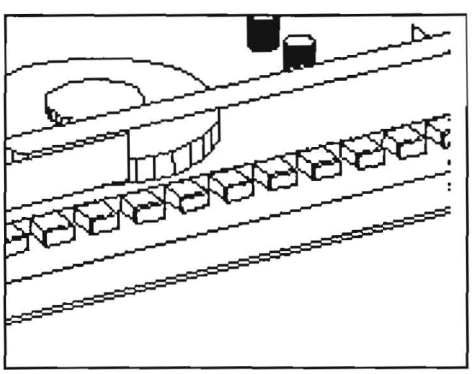
Ein anderes Beispiel: Die Lokalisation eines Objektes findet immer in Bezug zu dessen Oberobjekt oder zur Gesamtszene statt. Ein Kameraschwenk allein auf das Objekt würde unter Umständen den visuellen Bezug zum Umfeld zerstören, Die Kamera stünde danach zwar auf dem erwünschten Objekt, der Betrachter hätte jedoch den Überblick verloren und könnte das Kamerabild keiner Position im Raum mehr zuordnen.

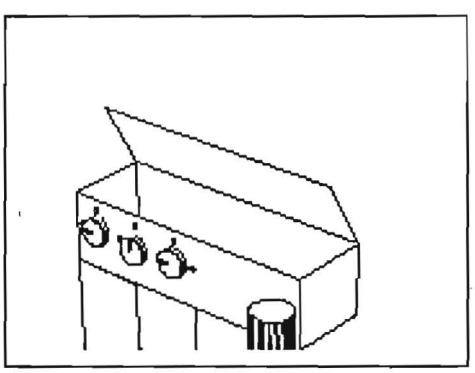
In Daumenkino 4 ist dieser Fall anhand der Lokalisation zweier LEDs innerhalb der gleichen LED-Reihe an der Frontseite eines Gerätes gezeigt. Zu Beginn der Animation ist die Gesamtszene sichtbar und ein Zoom auf die zu zeigende LED erfüllt das Visualisierungsziel. Würde die Kamera nun einfach mit festem Bildwinkel zur anderen LED schwenken, so wäre das Ergebnis ein sinnloses Vorbeirauschen von Details mit einer Endeinstellung, die dem Betrachter lediglich eine beliebige LED zeigt. Durch das Zurückfahren der Kamera auf das nächsthöhere Objekt in der Teil-von-Hierarchie, die LED-Reihe, wird jedoch der visuelle Kontext wieder klar, so daß die LED eindeutig als zweite von rechts identifizierbar ist. Die Dekompositionsregel zur Realisierung dieses Verfahrens heißt `localize-object` und findet sich ebenfalls in Anhang A.











4.4 Die Zwischensprache zur Beschreibung der Scripts

Um Animationen vom speziellen Grafiksystem unabhängig beschreiben zu können, braucht man eine neutrale Beschreibungssprache. Diese Sprache sollte einerseits die hierarchische Struktur des generierten Scriptes wiederspiegeln, damit nachvollziehbar bleibt, wie das Script entstanden ist, und um spätere Änderungen zu vereinfachen, andererseits aber sollte schon eine vollständige Beschreibung jedes einzelnen Bildes darin impliziert sein. Ein einfaches Mittel, beliebige Bäume (und die von BETTYs Planer generierten Scripte haben Baumform) darzustellen, sind vollständig geklammerte Ausdrücke oder in LISP-Terminologie geschachtelte Listen. An jedem Knoten des Baumes besteht die Möglichkeit, konkrete Angaben zu machen, wie z.B.

```
((:START-TIME   5.7)
 (:END-TIME     6.3)
 (:ANIM-OBJECT  CAMERA)
 (:MOTION-TYPE  MOVE)
 (:START-VALUE  (17 325 -39))
 (:END-VALUE    (201 325 70))
 (:NAME         "Move the camera"))
```

Hierbei bezeichnen die Angaben `:START-TIME` und `:END-TIME` absolute Zeitangaben in Sekunden, die einen genauen Zeitpunkt innerhalb der Animation beschreiben. Enthält der Knoten wie im obigen Beispiel die komplette Beschreibung einer Bewegung und kein Glied der Form

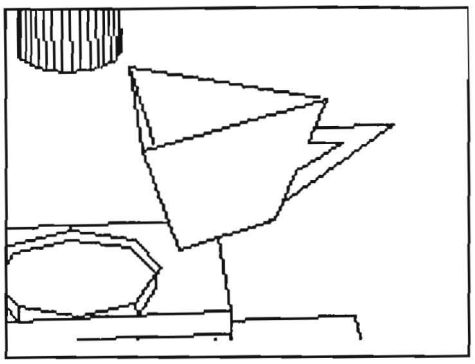
```
(:PARALLEL   ...)
(:SEQUENTIAL ...)
```

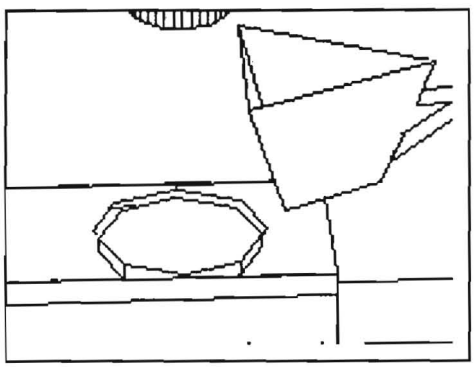
so handelt es sich um ein Blatt. Hat allerdings ein Glied des Knotens die Form

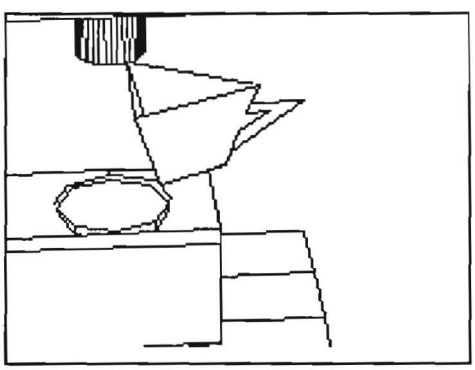
(:PARALLEL ($subs_1$) ... ($subs_n$)) bzw.
(:SEQUENTIAL ($subs_1$) ... ($subs_n$))

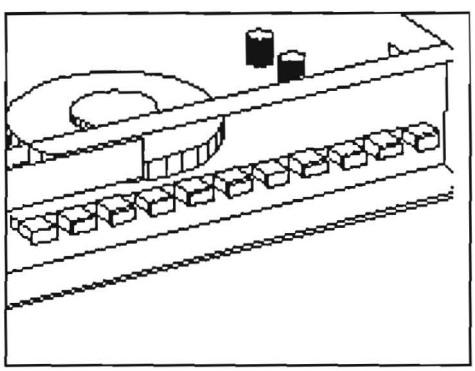
wobei $subs_i$ wieder ein Knoten ist, so handelt es sich um einen echten Knoten, d.h. das Script zerfällt an dieser Stelle in parallele bzw. aufeinanderfolgende Untersequenzen.

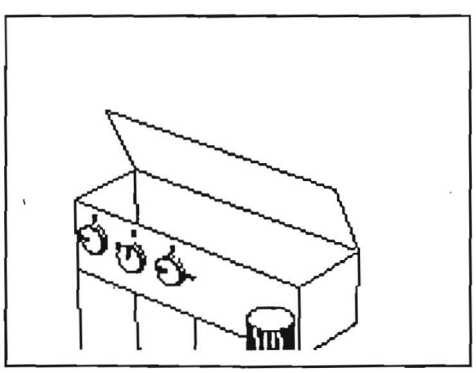
Außerdem besteht die Möglichkeit, konkrete Angaben auf Ebene jedes Knotens zu machen, die sich auf die darunterhängenden Knoten vererben. Soll beispielsweise die Kamera von X über Y nach Z bewegt werden, so läßt sich das durch folgenden Ausdruck beschreiben:











```
((:START-TIME   5.7)
 (:END-TIME     6.3)
 (:ANIM-OBJECT  CAMERA)
 (:MOTION-TYPE  MOVE)
 (:NAME         "Move the camera from X via Y to Z"))
 (:START-VALUE  (17 325 -39)) ; Punkt X
 (:END-VALUE    (201 325 70)) ; Punkt Z
 (:SEQUENTIAL
  ((:NAME         "Move the camera from X to Y"))
   (:END-TIME     6.0)
   (:END-VALUE    (201 325 -39))) ; Punkt Y
  ((:NAME         "Move the camera from Y to Z"))
   (:START-TIME   6.0)
   (:START-VALUE  (201 325 -39))) ; Punkt Y
```

Hierbei gelten die im Baum weiter oben spezifizierten Werte solange, bis in einem darunterhängenden Knoten speziellere Angaben gemacht werden. Am Beispiel: Die Startzeit 5.7 gilt sowohl für die gesamte oben beschriebene Bewegung, als auch für die erste Teilbewegung. In der zweiten Teilbewegung wird sie durch den spezielleren (weil tiefer im Baum angegebenen) Wert 6.0 ersetzt. Analoges gilt für die Endzeiten der Bewegungen sowie die Start- und Endwerte.

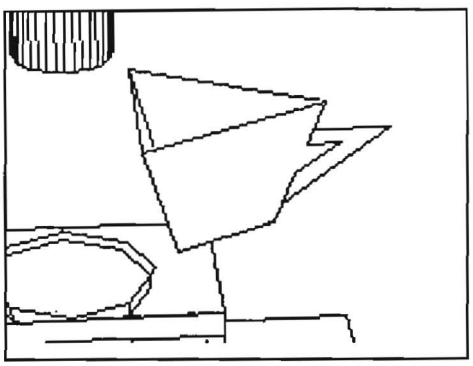
Somit sind alle Voraussetzungen geschaffen, eine Animation vollständig zu beschreiben, während ihr logischer Aufbau trotzdem nachvollziehbar bleibt. Einen Ausschnitt einer solchen Beschreibung zeigt auch Bild 4 auf Seite 19.

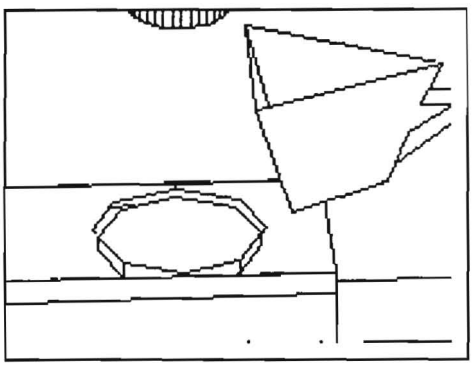
4.5 Die Schnittstelle zur Animationsrealisierung

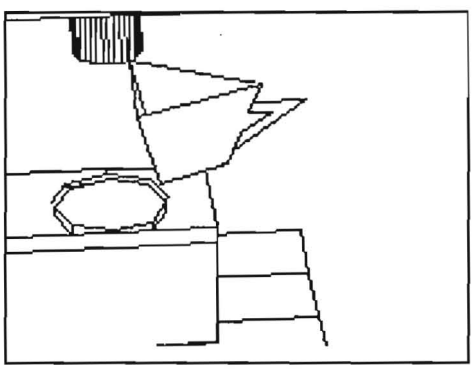
Die konkreten numerischen Werte, die zur vollständigen Beschreibung einer Animation auf unterster Ebene unumgänglich sind, werden bei der Realisierung der elementaren Aktionen von einer Schnittstellenkomponente, dem Backend berechnet und in die weiter oben erwähnten Schablonen eingesetzt.

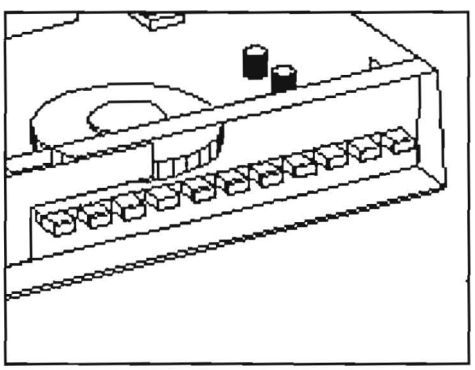
Das Backend selber stellt eine Sammlung solcher Routinen zur Umwandlung objektbezogener Angaben in absolute geometrische Werte und umgekehrt zur Verfügung. Hierzu werden teilweise Routinen aus dem Grafikpaket TOPAS ([Sc94]) benutzt, teilweise einfach Anfragen und Kommandos an die Ausgabekomponente weitergereicht und entsprechend aufbereitet (z.B. Festellen oder Verändern aktueller Objektpositionen).

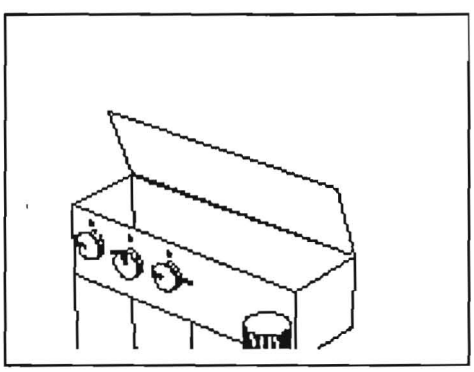
Schließlich übersetzt das Backend auch die Animationsbeschreibung aus der











eben beschriebenen Zwischensprache in die von der Ausgabekomponente benötigte Datenstruktur (z.B. Script-Datenstruktur bei S-Dynamics). Wird also das Ausgabemodul durch ein anderes ersetzt, so muß lediglich die Sammlung von Austausch- und Konversionsroutinen modifiziert werden, während der Planer selbst sowie das Regelwerk unverändert bleiben.

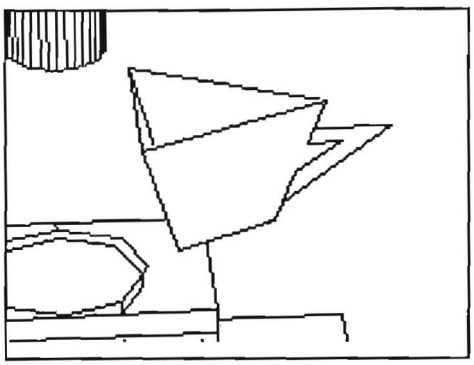
4.6 Die Animationsrealisierung

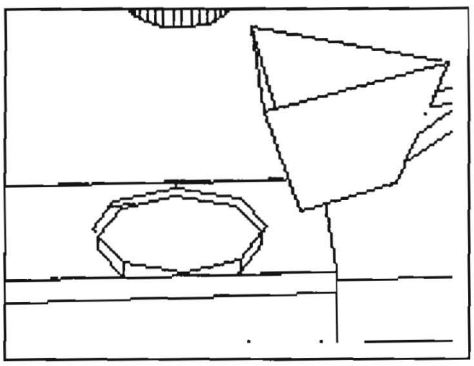
Die Berechnung der Einzelbilder der Animation geschieht in BETTY derzeit durch das Grafiksystem S-Geometry sowie das darin enthaltene Animationssystem S-Dynamics. Grund für diese Wahl war vor allem die konsistente Modellierung der Objekte. Die in WIP generierten technischen Grafiken stammen allesamt aus S-Geometry und benutzen die dort definierten Objekte und Bewegungstrajektorien. So war zur Generierung der Animationen keine zusätzliche Modellierung neuer Objekte oder Bewegungen nötig, da alles, was für die statischen Grafiken modelliert wurde, auch animiert werden kann.

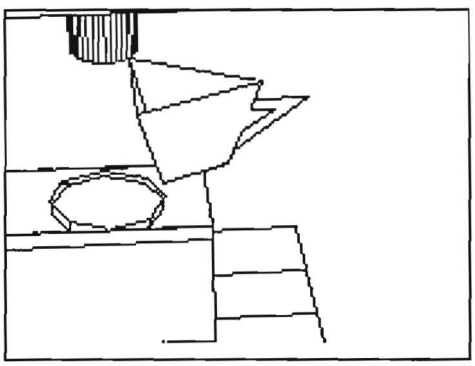
Dies hat den weiteren Vorteil, daß die von Grafik- und Animationsgenerierung erzeugten Präsentationen ein konsistentes Erscheinungsbild haben. Insbesondere sind die gezeigten Modelle identisch, aber auch Projektionsart, Strichstärke und andere äußere Merkmale sind zwischen verschiedenen Präsentationen konsistent. Das S-Dynamics-System bietet außerdem eine grafische Oberfläche zum Erstellen von Animationsscripten (Abb. 15), die bei einer automatischen Generierung die Möglichkeit bietet, diese Scripte nachzueditieren und das Ergebnis des Planungsprozesses zu überprüfen. Die erzeugten Animationen können abgespielt, modifiziert, neuberechnet oder bildweise angeschaut werden, was bei der Formulierung der Dekompositionsregeln eine große Hilfe war.

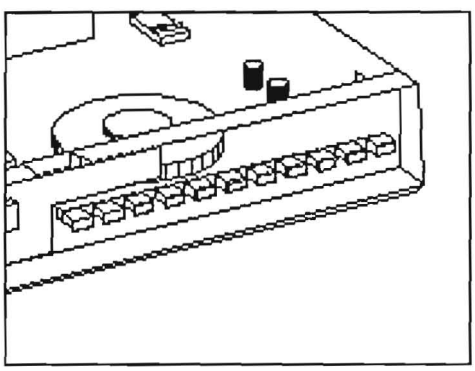
4.7 Aus TOPAS stammende Systemfunktionen

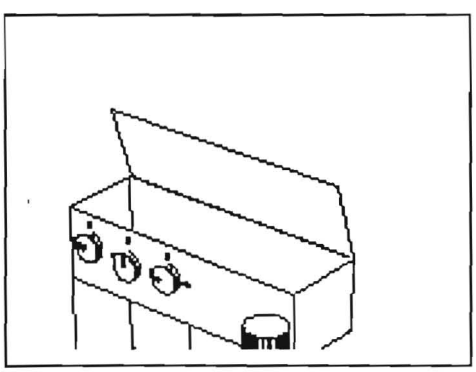
Das Grafikrealisierungs-System TOPAS , das in [Sc94] eingehend beschrieben wird, stellt eine umfassende Sammlung von Funktionen zur automatischen und halbautomatischen Erstellung technischer Grafiken dar. Es benutzt als Ausgabekomponente ebenfalls das S-Geometry-System und stellt komplexe Funktionen wie automatische Perspektivenwahl, Berechnung von Explosionszeichnungen und die Verwaltung von Weltszenen in der Modellwelt sowie illustratorischer Szenen zur Verfügung.











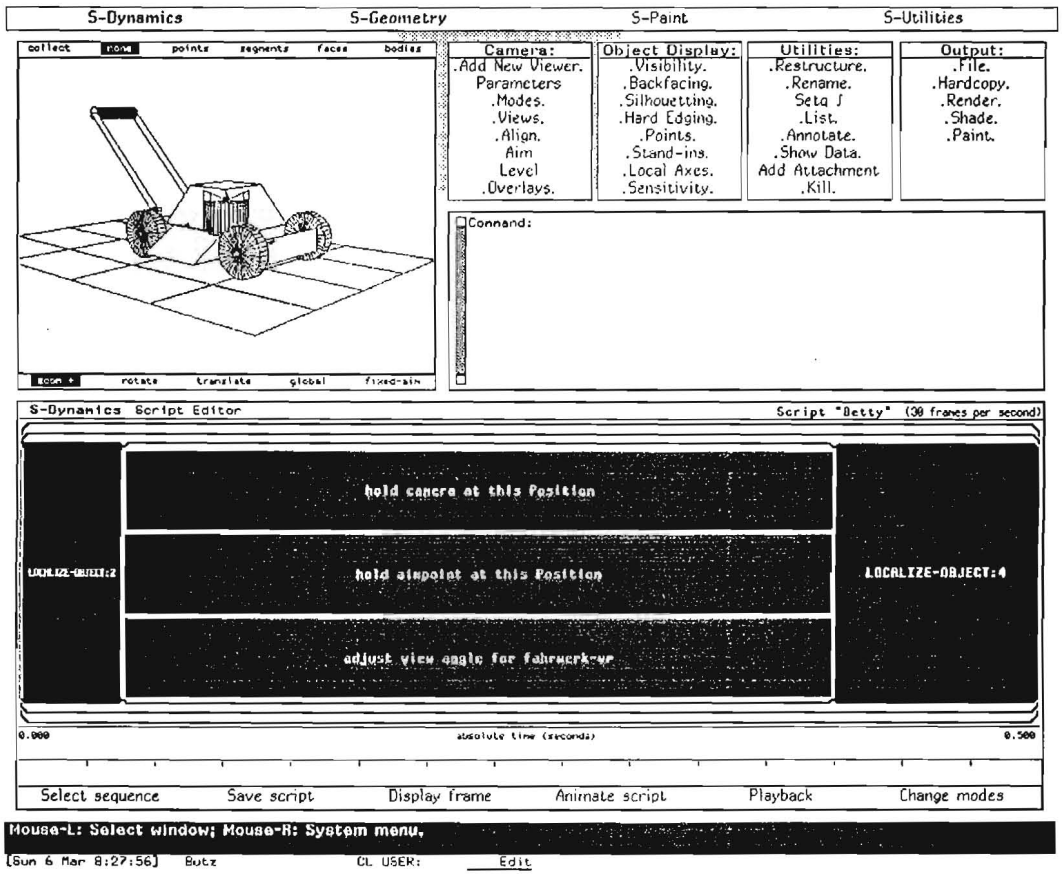


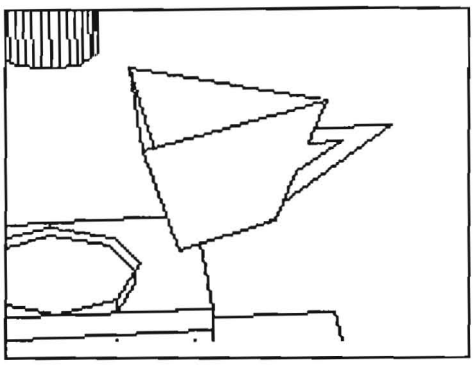
Abbildung 15: Grafische Oberfläche der Ausgabekomponente

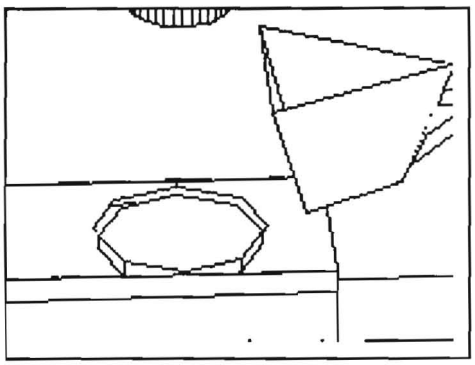
4.7.1 Perspektivenwahl

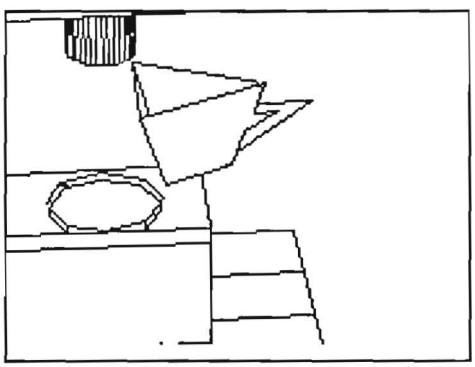
Um ein Objekt in einer Animation sichtbar zu machen, muß sich die Kamera an einer geeigneten Position befinden, auf einen geeigneten Punkt gerichtet sein und außerdem die Brennweite des Objektivs derart eingestellt sein, daß das Objekt in vernünftiger Größe im Bild erscheint. Die Werte für Kameraposition und Kamerazielpunkt werden bei vorgegebener Objektivbrennweite von TOPAS automatisch derart bestimmt, daß das Objekt unter einer Reihe von Nebenbedingungen richtig im Bild erscheint.

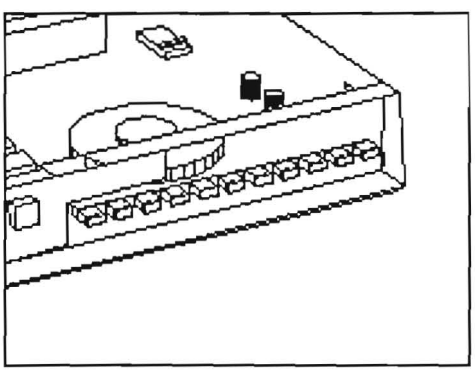
Solche Nebenbedingungen sind z.B., welches Objekt das Zentrum des Bildes bilden soll, welche Objekte außerdem sichtbar sein sollen und von welcher Seite das Objekt gezeigt wird (vgl. [An90]).

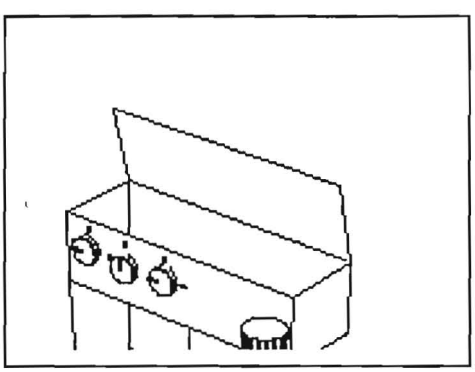
Dieser Ansatz läßt sich folgendermaßen auf die automatische Generierung











von Animationen übertragen: In der Regel bietet die Wahl eines guten Kamerastandortes für eine Fotografie auch gleichzeitig einen guten Ausgangspunkt für eine Filmeinstellung. Bei der Animation wie beim Film muß jedoch darüberhinaus noch die eventuelle Veränderung der Objektpositionen mit berücksichtigt werden. Objekte, die sich in der Einstellung bewegen, dürfen nicht plötzlich aus dem Bild verschwinden oder durch andere Objekte verdeckt werden.

Um dies zu gewährleisten, wurden in BETTY aufgrund der Trajektorien, die die auftretenden Bewegungen beschreiben, zusätzliche Nebenbedingungen aufgenommen, unter denen wiederum der gleiche Mechanismus zur Perspektivenwahl angewandt werden kann, der in TOPAS für statische Grafiken existiert. Auch im Interesse der Effizienz (Vermeidung von Mehrfachimplementationen) und der Konsistenz (Kriterien für die Perspektivenwahl) lag es deshalb nahe, TOPAS für diese Funktionen zu verwenden.

4.7.2 Berechnung der Explosion

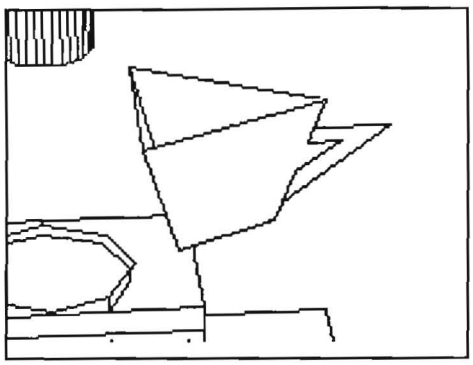
Eine weitere Funktion des Grafikdesign-Systems ist die automatische Erstellung von Explosionszeichnungen. Hierbei werden unter Berücksichtigung der Kameraposition am 3D-Modell die Bewegungen der einzelnen zu explodierenden Teile entlang ihrer Hauptachsen berechnet, die bei fester Kameraeinstellung eine überdeckungsfreie Darstellung im Explosionsschaubild ergeben.

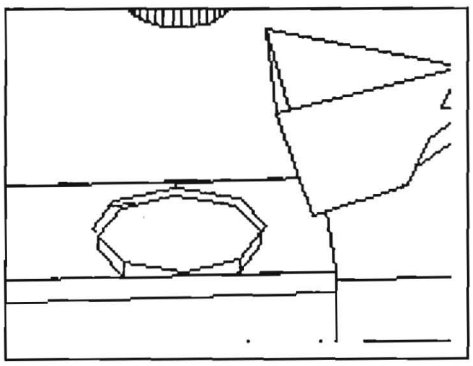
Auch diese illustratorische Technik läßt sich animieren. Hierzu wird einfach die Bewegung der einzelnen Teile aus ihrer normalen Position in die explodierte Position animiert dargestellt. Das Ergebnis ist eine Sequenz, die eine langsame Auseinanderbewegung der Objekte zeigt, so daß für jedes Teil einzeln erkennbar bleibt, wo es sich ursprünglich befand, und wie es an seine Endposition kommt.

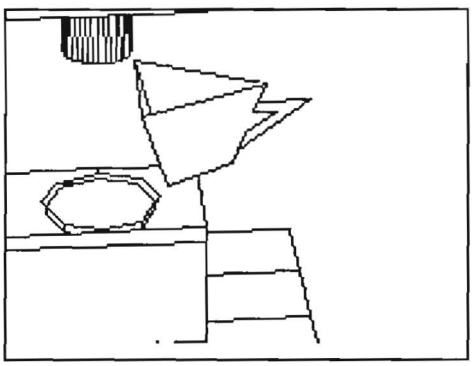
Eine getrennte Visualiusierung der Bewegungen und der ursprünglichen Objektpositionen durch Hilfslinien kann also entfallen, da diese Information in einer Animation z.T. in die Dimension Zeit projiziert werden kann (siehe hierzu Kap. 2.2.1). Die animierte Explosion stellt also einen sehr direkten Weg dar, den Aufbau und räumlichen Zusammenhang komplexerer Objektgruppen zu visualisieren. Ein Beispiel hierzu zeigt die Bildfolge 18.

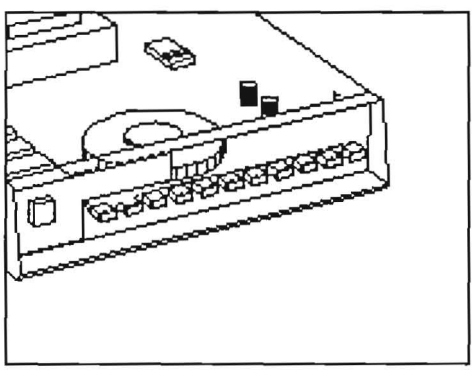
4.7.3 Szenen, Objekte und Bewegungen

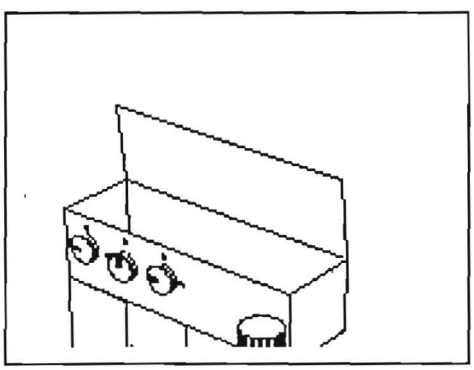
Schließlich stellt das TOPAS -System eine komplette Verwaltung von Weltszenen der modellierten geometrischen Welt zur Verfügung. (Auch eine Verwaltung der darauf aufsetzenden illustratorischen Szenen ist möglich, soll hier jedoch nicht











näher erläutert werden.) Die Beschreibung einer Weltszene enthält Angaben über die in der Szene enthaltenen Objekte, über ihre Bewegungszustände und Eigenschaften.

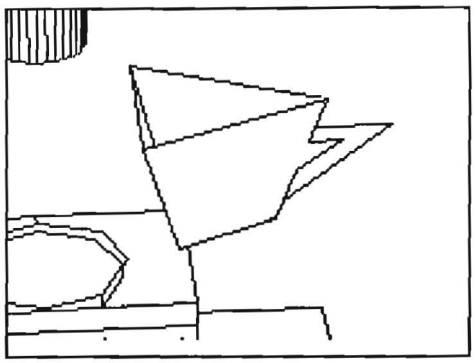
Zu den Objekten selbst liefert TOPAS Angaben über ihre räumlichen Ausdehnungen, ihren geometrischen Mittelpunkt, sowie über das Vorhandensein bestimmter Ausrichtungen. So ist es z.B. möglich, die Vorderseite eines Elementes herauszufinden, sofern eine solche existiert, oder die Kamera auf den Mittelpunkt eines Objektes zu richten. Wie diese Berechnungen im einzelnen ablaufen, ist in [Sc94] eingehend erläutert.

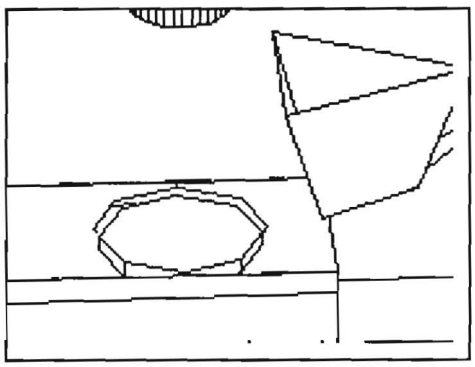
Die Bewegungen schließlich, die die in der Modellwelt enthaltenen Objekte ausführen können, sind in einer Datenstruktur modelliert, die gleichzeitig die Visualisierung dieser Bewegungen durch Animation und die Darstellung in statischen Grafiken (durch Pfeile, ghost-images, ...) ermöglicht. Hierzu werden bestimmte Charakteristika der Bewegung abgelegt, wie z.B., ob es sich um eine Translation, Rotation oder Trajektorienbewegung handelt, ferner ein Bewegungsbetrag zur Rotation und Translation, eine Rotationsachse, ein Mittelpunkt und ein Verweis auf die im Grafiksystem modellierte Bewegungstrajektorie, sofern die jeweiligen Angaben existieren. Eine solche Bewegungsbeschreibung ist immer spezifisch für ein Objekt. Sie ist für dieses Objekt außerdem eindeutig charakterisiert durch einen Namen. Zwei Beispiele:

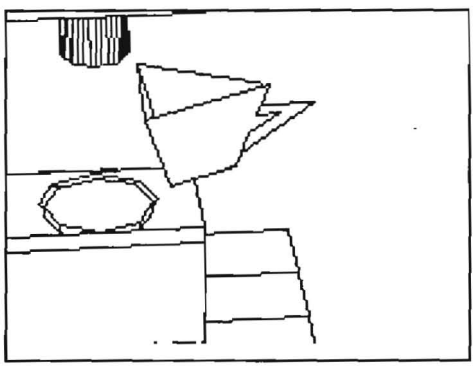
```
(create-movement
 :motionname "wegnehmen"
 :objectname "tasse-1"
 :motiontype ':trajectory
 :trajectory '((0 0 0) (20 30 0) (100 20 0) (130 -83 0)))
```

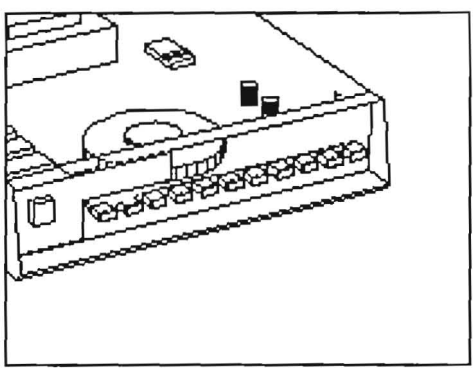
beschreibt die Bewegung zum Wegnehmen einer modellierten Tasse unter einer Espressomaschine (Daumenkinos 1-3). Die Bewegung ist eine Trajektorienbewegung und durch einen Spline mit 4 Stützpunkten explizit beschrieben.

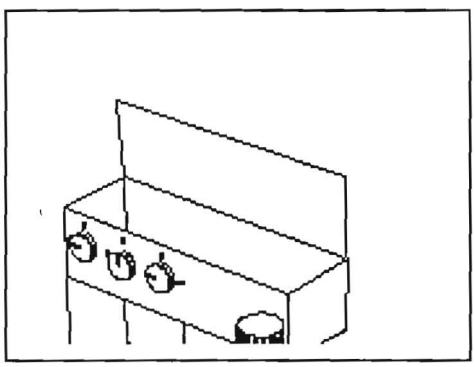
```
(create-movement
 :motionname "aufklappen"
 :objectname "oberteilkasten-1-deckel"
 :motiontype ':rotate
 :axis       'x
 :amount     90
 :centerpoint '(125 290 -80))
```





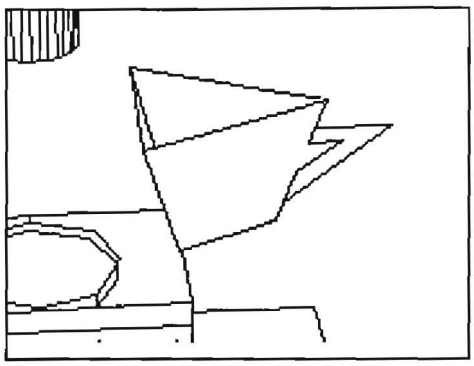


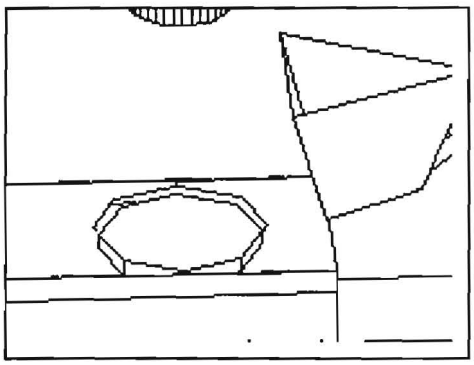


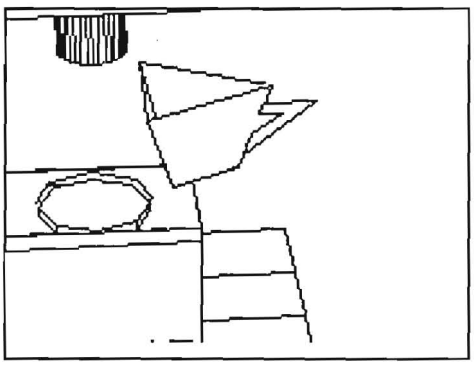


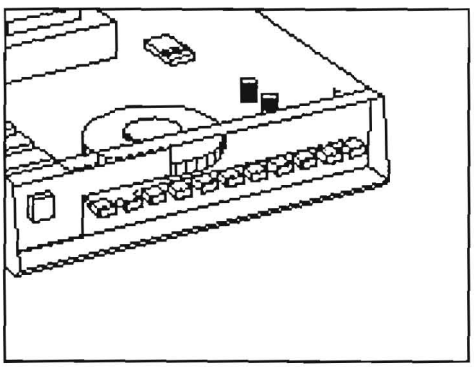
beschreibt das Aufklappen eines Deckels (Daumenkino 5, Bild 31 - 60) als Rotation eines Objektes um einen bestimmten Betrag und um eine bestimmte Achse durch einen bestimmten Punkt. Trajektorien sind dabei generell als Splines durch beliebig viele Stützpunkte beschrieben und auch in S-Geometry so modelliert.

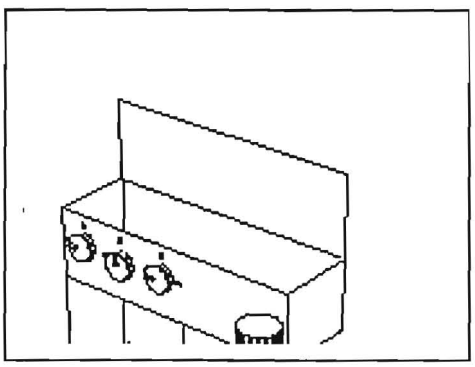
Diese Beschreibung der Objektbewegungen wurde gemeinsam mit dem Autor des TOPAS -Systems erarbeitet, um von vorneherein die mehrfache Verwendbarkeit der Darstellung und die Ableitbarkeit der verschiedenen Präsentationen sicherzustellen. Sie bildet eine Minimallösung hinsichtlich der zu spezifizierenden Angaben, die trotzdem den von beiden Seiten gestellten Anforderungen genügt.











5 Implementation des Systems

5.1 Sprache, Stil, Umfang

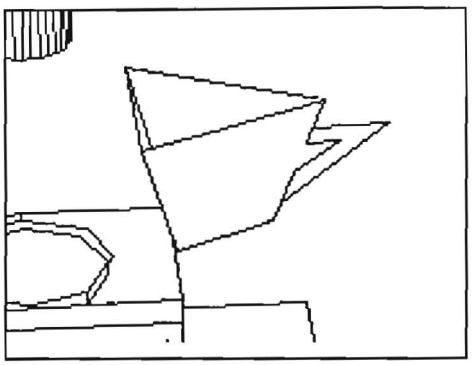
Das System BETTY ist (wie das WIP Gesamtsystem) in Symbolics Common Lisp implementiert. Dieser LISP-Dialekt bietet umfangreiche Möglichkeiten zur objektorientierten Programmierung (Flavor System) sowie zur komfortablen Programmierung grafischer Ausgaben (window System), die in ihrem Funktionsumfang über die ebenfalls enthaltenen Pakete CLOS und CLIM hinausgehen.

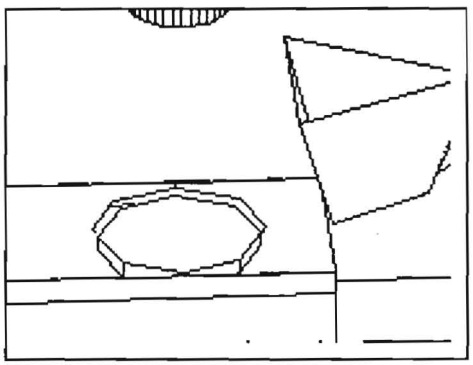
BETTY ist objektorientiert implementiert, das heißt es gibt eine Objektklasse `animation-planner`, deren Instanzen jeweils einen kompletten Animationsplaner darstellen. Diesem Planer werden die Visualisierungsziele übergeben, und er erzeugt das Animationsscript, das vom S-Dynamics-System in eine fertige Animation umgerechnet wird. So ist es möglich, mehrere Animationsplaner nebeneinander zu betreiben, die bei einer parallelen Vorgehensweise (entsprechende Systemarchitektur vorausgesetzt) ihre Arbeit unabhängig voneinander erledigen und so mehrere in der gleichen Präsentation benötigte Animationen parallel generieren.

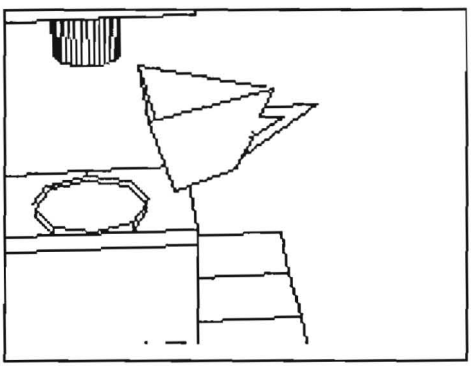
Der Codeumfang des BETTY -Systems alleine beträgt etwa 130 Kbytes. Hinzu kommen nochmals etwa 1,38 Mbytes aus im WIP - System schon vorhandenen Komponenten wie z.B. TOPAS . Das Regelwerk umfaßt 49 Dekompositionsregeln und ist im Anhag A komplett aufgelistet und kommentiert.

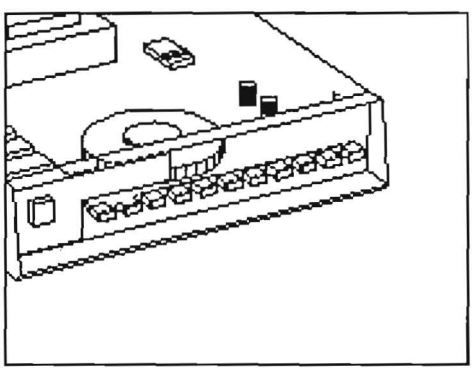
5.2 Portierbarkeit

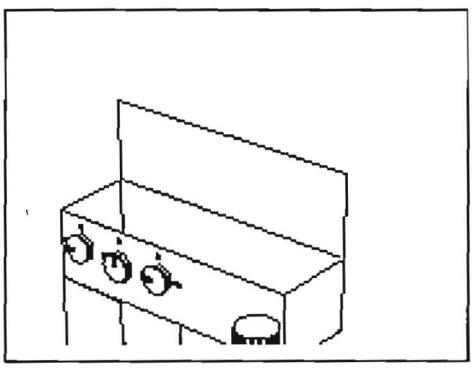
Das System in seiner derzeitigen Form läuft auf einer Symbolics XL1200 unter Genera 8.0. Eine Ausführung auf allen Symbolics- Architekturen ist prinzipiell möglich, jedoch vom Vorhandensein des S-Dynamics-Systems und des nötigen Speicherplatzes abhängig. So ist die Ausführung auf einem MAC-Ivory-board oder einem UX-400 oder UX-1200 Unix-coprozessor-board zwar prinzipiell möglich, wegen einer wichtigen Einschränkung jedoch kaum sinnvoll: Die Grafikausgabe des S-Dynamics-Systems arbeitet mit der X-Windows-Schnittstelle der UX-boards nicht zusammen, so daß dort zwar eine Berechnung prinzipiell möglich wäre, die Ausgabe jedoch unmöglich ist. Für die MAC-Ivory-boards trifft diese Beschränkung zwar nicht zu, jedoch sind sie von ihrer Rechenleistung her nicht mit der XL1200 zu vergleichen und erschweren die Arbeit durch unangenehm lange Rechenzeiten.











Das Kernstück des Systems, der Planer selbst, ist in reinem Common LISP geschrieben und sollte daher auf jedem Common-LISP System laufen. Die Vorführumgebung, die ja außerhalb der eigentlichen Animationsplanung und Generierung steht und die einzige direkte grafische Schnittstelle des Systems bildet, ist so weit in die GENERA-Oberfläche integriert, daß eine direkte Ausführung in anderen LISP-Umgebungen nicht möglich ist.

5.3 Austauschbarkeit der Komponenten

Wie eingangs schon gesagt, zerfällt das System BETTY in zwei grundlegende Bestandteile, die Animationsplanung und die Animationsrealisierung.

5.3.1 Realisierungskomponente

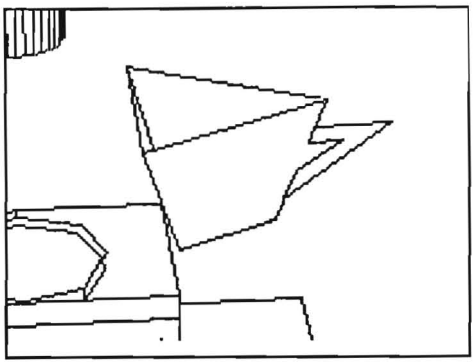
Die vom Planer erzeugten Scripts können von verschiedenen Animations- und Grafiksystemen in animierte Bildfolgen übersetzt werden, denn das Script selbst enthält keinerlei Angaben über Maschinen- oder systemspezifische Parameter wie Bildfrequenz, Grafikauflösung, Farbtiefe oder ähnliches. Es werden lediglich Objektbewegungen in der Zeit angegeben, die von einer konkreten Realisierung unabhängig sind. Dies ist ein entscheidender Vorteil des Systems gegenüber einem Planungsansatz, der die Details der Animation bis auf einzelne Bilder festlegt.

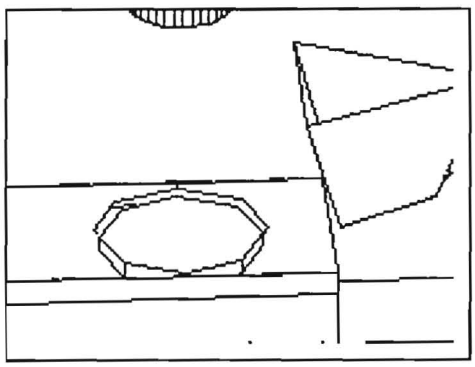
Die Realisierung der Animationen kann also je nach Bedarf oder Verwendungszweck in völlig unterschiedlicher Art ausfallen. Denkbar sind verschiedene Qualitätsstufen der Ausgabe von photorealistisch gerenderten Animationen, die auf Videorecorder oder Bildplatte ausgegeben werden können, bis zu Realzeit-Lösungen, die die generierten Scripts inkrementell schon zum Planungszeitpunkt in Animationen umsetzen und so eine sehr direkte und kurzfristige Präsentation ermöglichen, wie dies z.B. in Steuersystemen und technischen Leitständen notwendig ist.

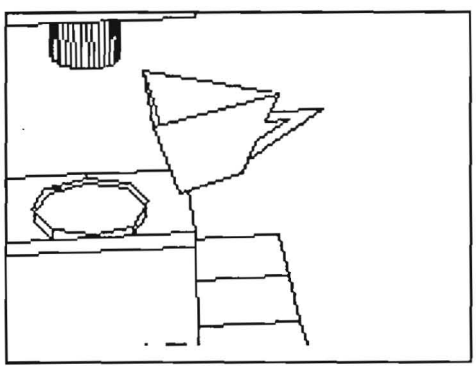
All diese Variationen setzen lediglich einen Austausch der Realisierungskomponente, d.h. des Grafiksystems sowie der Schnittstellenkomponente zu dessen Ansteuerung voraus. Alle weiteren Programmteile bleiben unberührt.

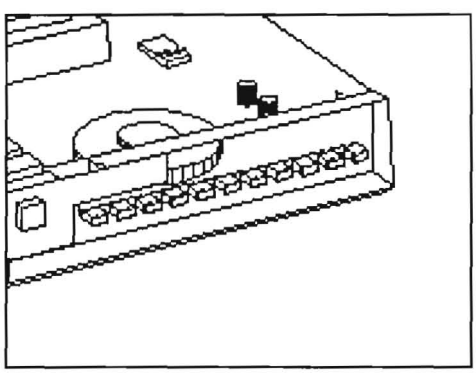
5.3.2 Planungskomponente

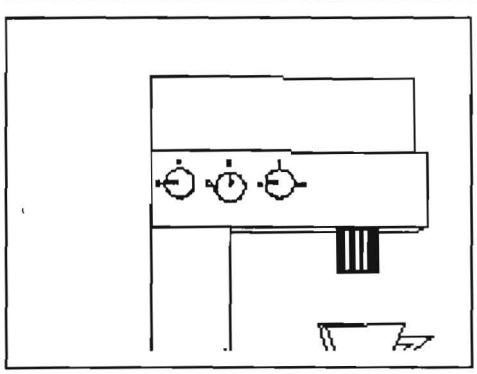
Die Planungskomponente selbst ist lediglich durch ihre Ein- und Ausgabespezifikation festgelegt: Am Eingang werden die Visualisierungsziele übergeben, am











Ausgang kommt ein Animationsscript heraus. Wie dieses Script im Einzelnen erzeugt wird, berührt die anderen Teile des Systems nicht, solange die Schnittstellen unverändert bleiben.

Grundsätzlich ist es denkbar, das BETTY System an dieser Stelle durch andere Planungsalgorithmen zu erweitern, bzw. mehrere verschiedene Verfahren z.B. je nach verfügbarer Planungszeit zur Verfügung zu stellen. Obwohl die hier vorgestellte hierarchische Planungsvariante ihre Tauglichkeit unter Beweis gestellt hat, wäre es sicherlich interessant, andere Varianten zu implementieren und an dieser Stelle auszuprobieren.

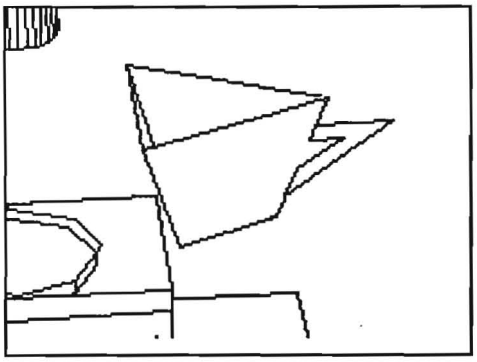
Jeder Wechsel des Planers jedoch bleibt ohne Einfluß auf die anderen Programmteile, solange die Syntax und Semantik der Schnittstellen eingehalten wird.

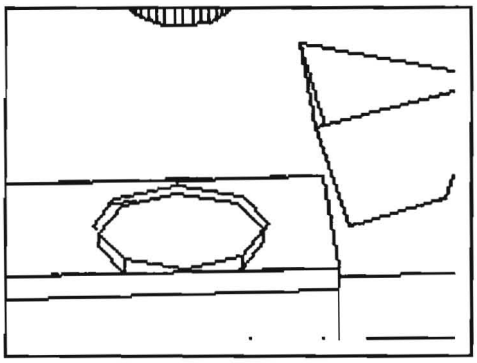
5.3.3 Regelwerk

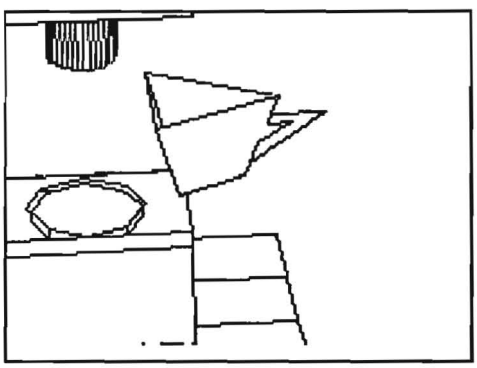
Das Wesen eines Regelwerkes in regelbasierten Systemen ist es, austauschbar und erweiterbar zu sein. So bildet das vorliegende Regelwerk lediglich eine Lösung für die gestellten Aufgaben, die mit der Zeit verändert und perfektioniert wurde. Die Regelmenge ist in dem Sinne statisch, daß das System sie nicht selbständig verändert oder erweitert. Sie ist in dem Sinne offen, daß es möglich ist, durch bloßes Hinzufügen von Regeln die Menge der realisierbaren Visualisierungsziele zu erweitern.

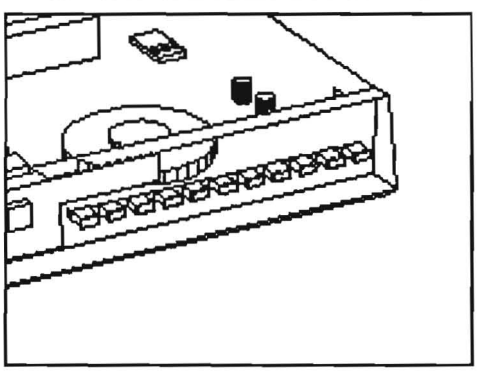
Die vorliegenden 49 Regeln bilden lediglich eine leicht überschaubare Musterlösung und insofern einen konstruktiven Beweis für die Tauglichkeit des gesamten Systems, sie bilden jedoch in keiner Hinsicht eine Obergrenze, was die Leistungsfähigkeit und Flexibilität des Planungsansatzes betrifft. So könnte man sich beispielsweise die Codierung weiterer Visualisierungsziele mit mehreren kontextabhängigen Regeln vorstellen oder die Nutzung des Konzeptes rekursiver Regeln, wie es in Kap. 4.3.3 erläutert wird.

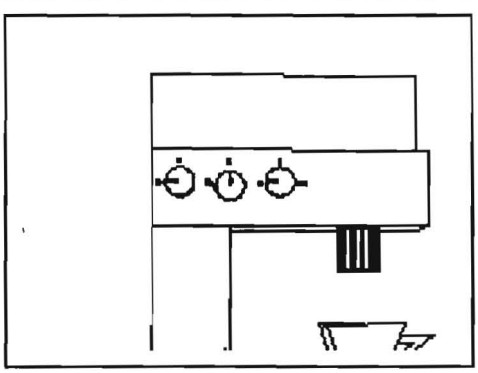
Selbstverständlich bleiben bei einer Erweiterung des Regelwerkes alle anderen Systemteile unberührt, außer daß sich die Menge der vom Planer akzeptierten Visualisierungsziele sowie die Form ihrer Realisierung mit den Regeln ändert.











6 Funktionsumfang des BETTY-Systems

6.1 Die erzeugten Animationen

Da die Ergebnisse eines Animationssystems in seinen Animationen bestehen und diese einen Vorgang in der Zeit darstellen, der auf gedrucktem Papier wiederum nur beschränkt darstellbar ist, habe ich für die vorliegende Arbeit zwei verschiedene Arten der drucktechnischen Darstellung von Animationen verwandt.

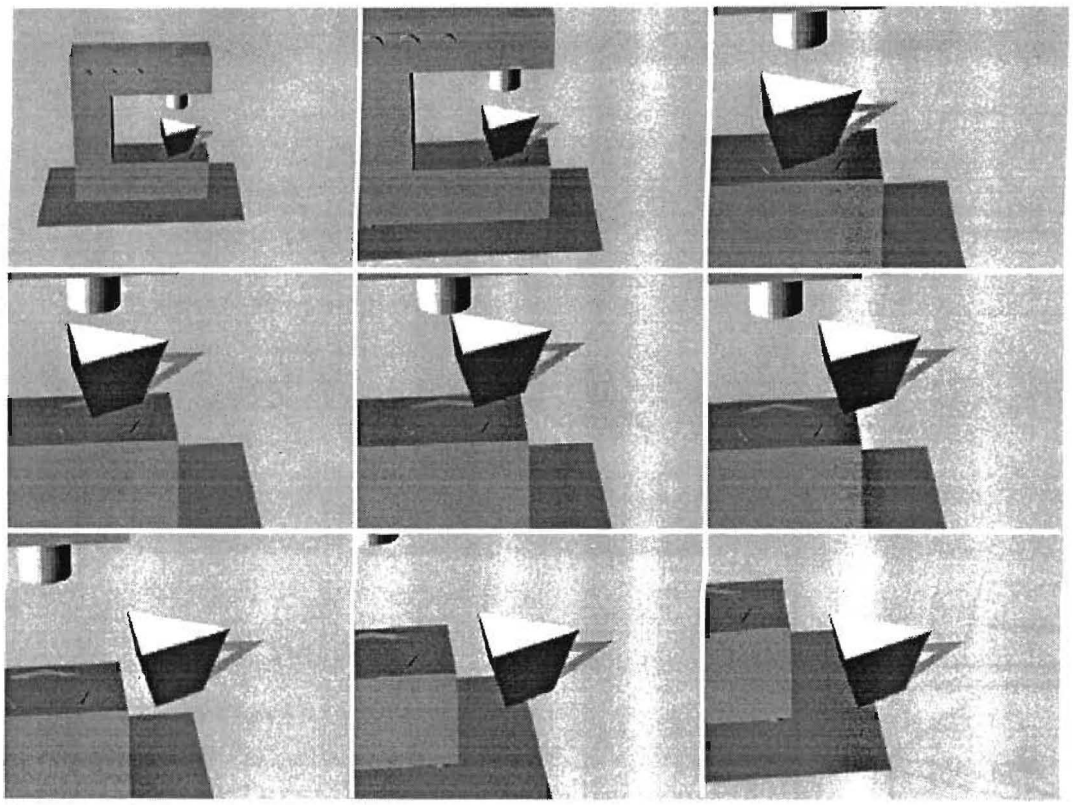
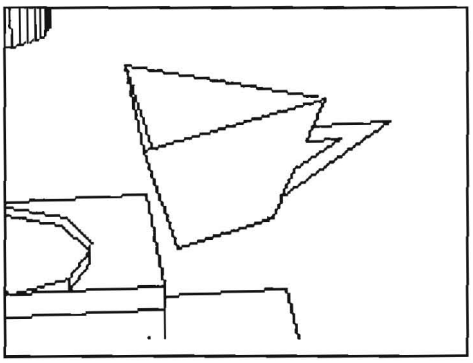
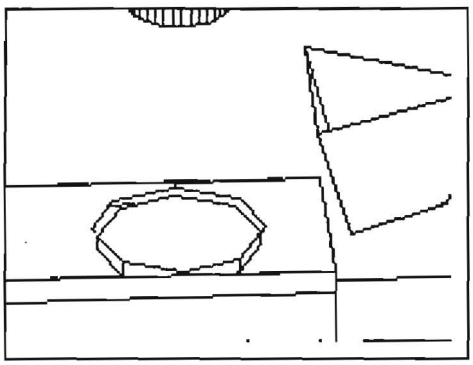
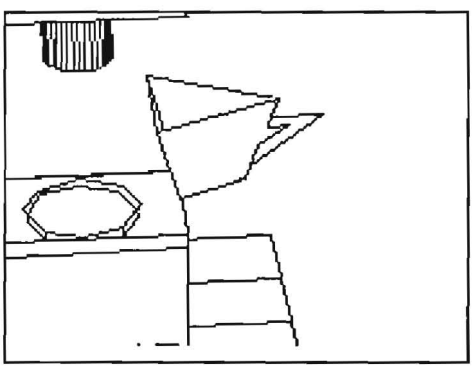


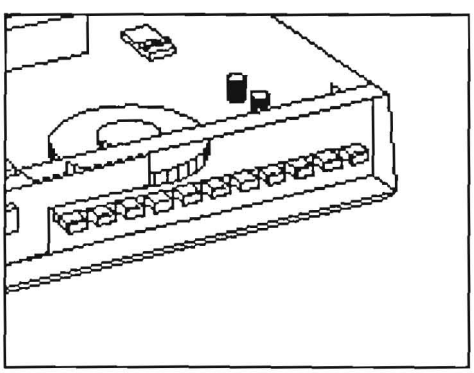
Abbildung 16: Zeigen einer Bewegung

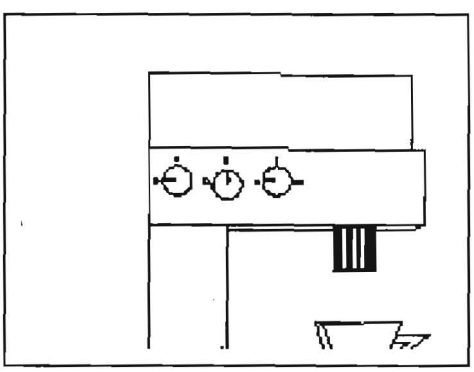
Einerseits bilden die Daumenkinos auf den Rückseiten der Blätter dieses Bandes (Bedienung siehe Einleitung) Beispiele für Animationen, die mit BETTY erzeugt wurden, andererseits sind aber drei Beispielanimationen (Bild 16 - 18) auch als Folge von Schnappschüssen ausgedruckt und zeigen eine Variante der Animationsrealisierung mit Licht und Schatten. Die Bilder sind zeilenweise von links











oben nach rechts unten zu betrachten.

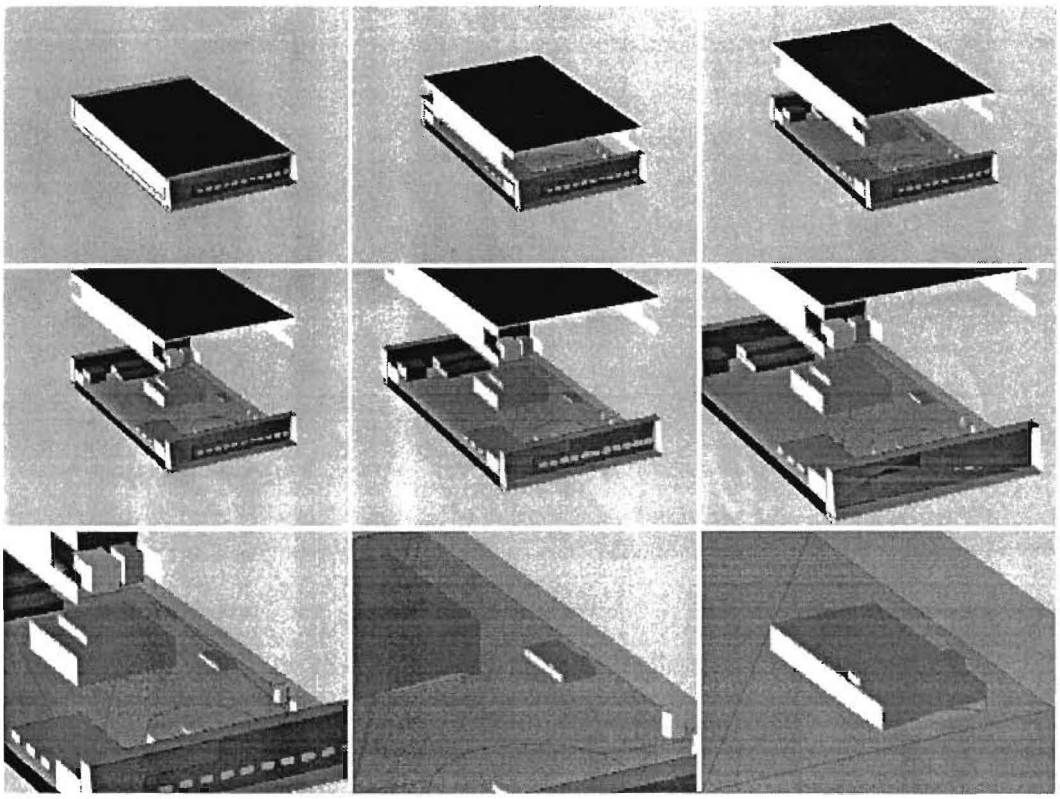
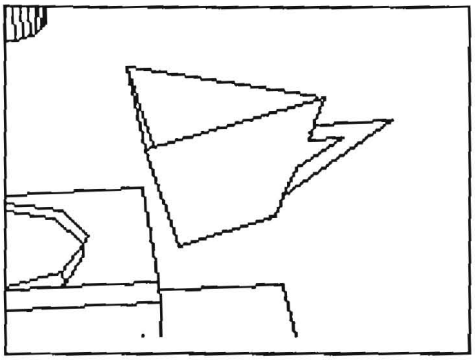


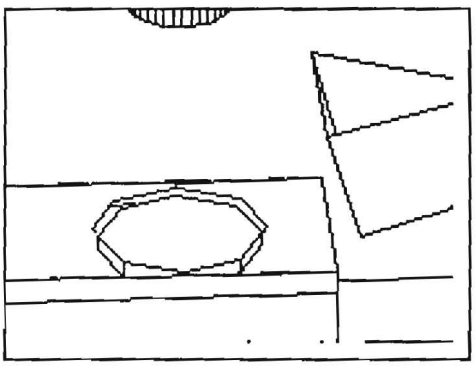
Abbildung 17: Lokalisation eines Teiles

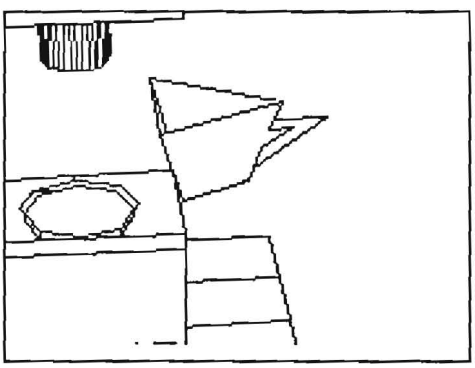
6.2 Die unterstützten Visualisierungsziele

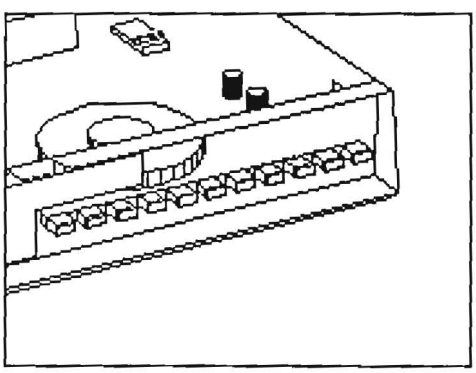
Die derzeit unterstützten (und vom Regelwerk beschriebenen) Visualisierungsziele sind im einzelnen:

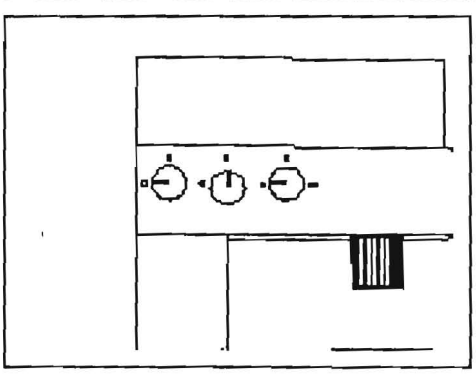
- Lokalisation von Objekten
- Zeigen der Bestandteile von zusammengesetzten Objekten
- Darstellung von Objektbewegungen











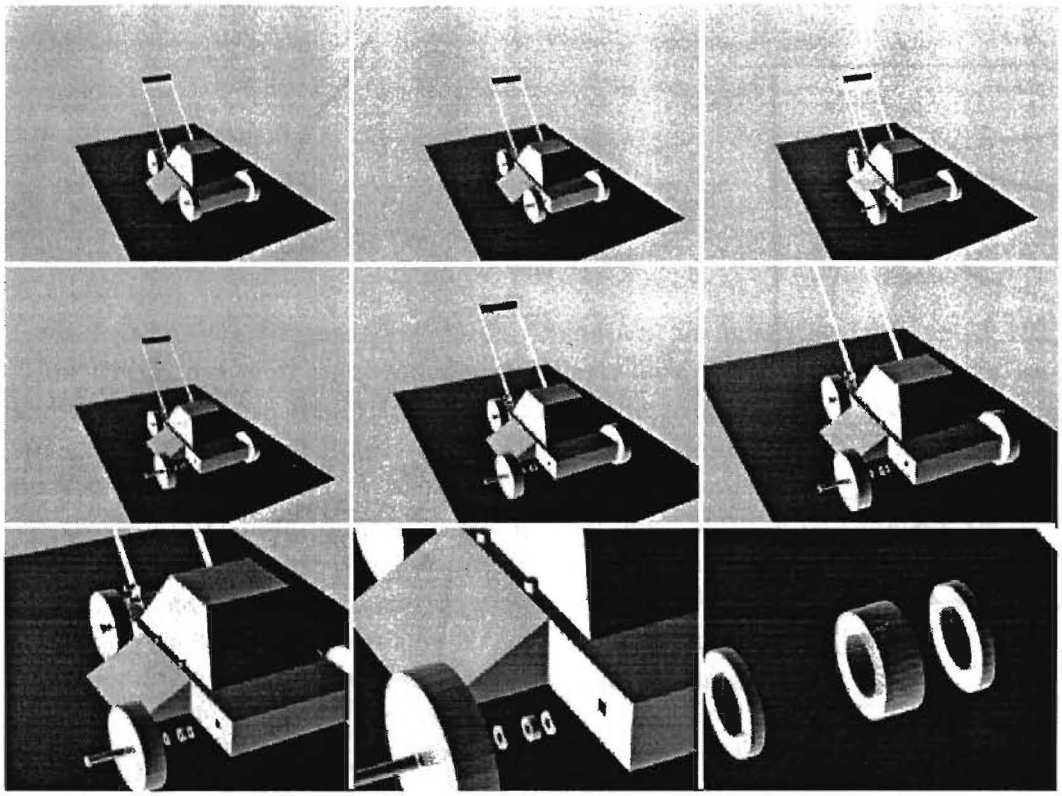
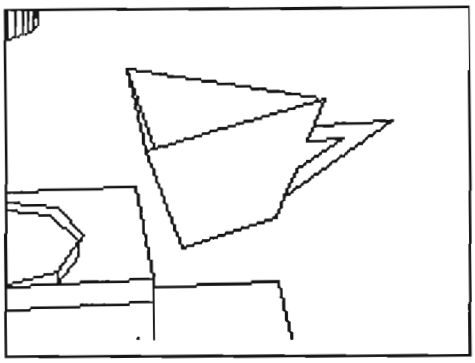


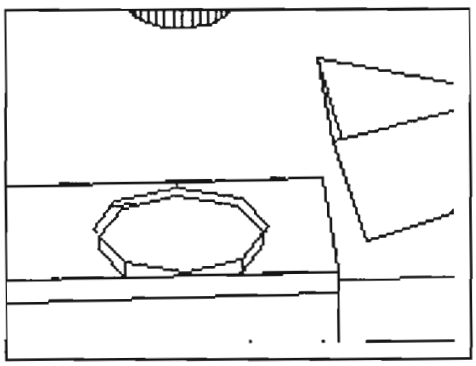
Abbildung 18: Zeigen der Bestandteile einer Baugruppe

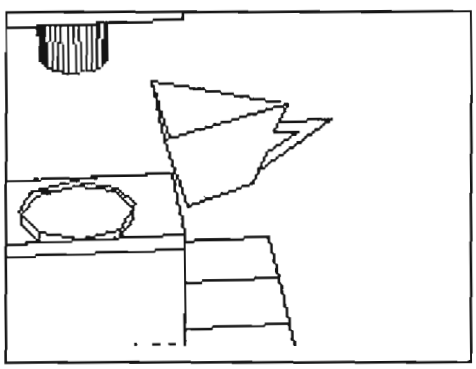
Die Lokalisation ist ein Standardelement bei der Generierung von Bedienungsanleitungen am Bildschirm. Sie erfüllt eine Funktion, die in natürlicher Sprache etwa durch den Satz „Der Schalter befindet sich unter der Abdeckung rechts auf der Platine, gleich neben dem Transformator."erfüllt wird: Beim Betrachter der Präsentation wird eine Vorstellung über die Position des Objektes im Raum sowie über seine räumlichen Beziehungen zu anderen Objekten erzeugt.

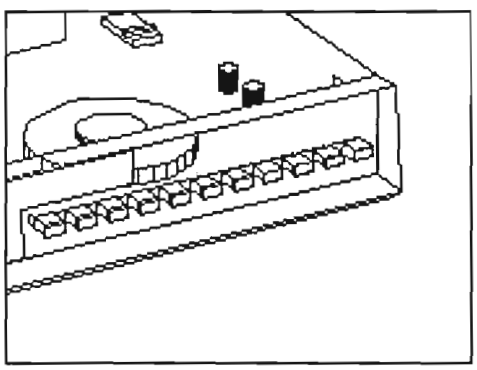
Das Zeigen der Bestandteile eines zusammengesetzten Objektes oder einer Baugruppe geschieht durch einen langsamen Übergang der Baugruppe aus ihrem ursprünglichen Zustand in eine Explosionsdarstellung der betreffenden Teile, wie man sie aus technischen Zeichnungen kennt. Die Explosion dient dazu, beim Betrachter eine Vorstellung zu erzeugen, wie eine Objektgruppe räumlich zusammenhängt, in welcher Relation ihre Teile zueinander stehen und welches jeweils die einzelnen Bestandteile der Baugruppen sind. Diese Sachverhalte in natürlicher Sprache auszudrücken, ist oft fast unmöglich, da die Anzahl der räumlichen Relationen, die ausgedrückt werden, sehr groß ist.

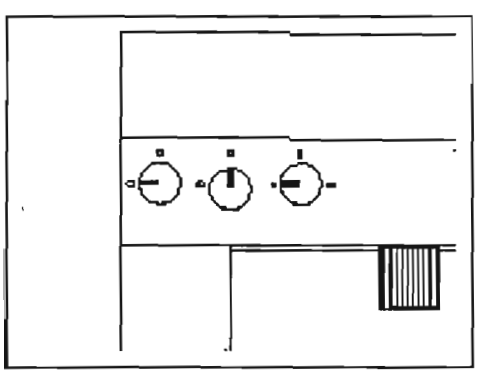
Die Darstellung von Objektbewegungen schließlich war die Funktion, die den











Gedanken an eine Präsentation durch Animation am nächsten legte (Bild 16). Beim Betrachter wird eine sehr direkte Vorstellung davon erzeugt, wie ein bestimmter Vorgang in Raum und Zeit abläuft. Hierzu sind in der Animation keine sprachlichen Umschreibungen oder metagrafischen Elemente notwendig. Ein Satz der Art „Drehen Sie den mittleren der drei Regler um 90 Grad nach rechts auf die zweite Markierung, öffnen Sie sodann die obere Klappe und drehen Sie den mittleren Regler danach wieder um 90 Grad nach links in die Ausgangsposition zurück!"ist bei weitem nicht so direkt, wie eine Animation, die den gleichen Vorgang visualisiert (Daumenkino Nr.5). Die Präsentation einer Bewegung durch eine Animation ist die direkteste Art ihrer Visualisierung.

6.3 Die erzeugten Scripts

Die vom Animationsplaner erzeugten Scripts können nahezu beliebig lang und beliebig komplex werden, je nachdem wie umfangreich das Regelwerk ist und wieviele einzelne Visualisierungsziele in eine Animation gepackt werden.

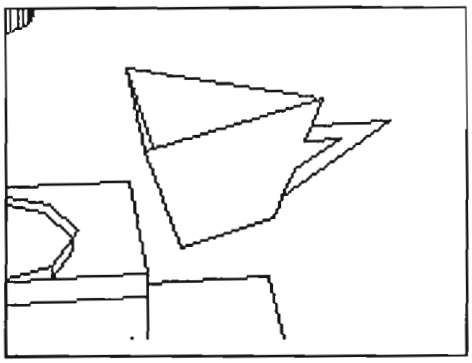
Als Anhaltspunkt für die Komplexität der Scriptbäume betrachte man beispielsweise das Script zur Visualisierung einer Objektbewegung, das Daumenkino Nr.3 zugrunde liegt. Dieses Script enthält 389 elementare Bewegungen, die in einem Baum von 14 Hierarchieebenen angeordnet sind. Das Script wird in 6 Sekunden einschließlich Perspektivenwahl und geometrischer Berechnungen geplant und innerhalb 50 Sekunden in eine Animation der Länge 5 Sekunden bei 15 Bildern pro Sekunde umgerechnet.

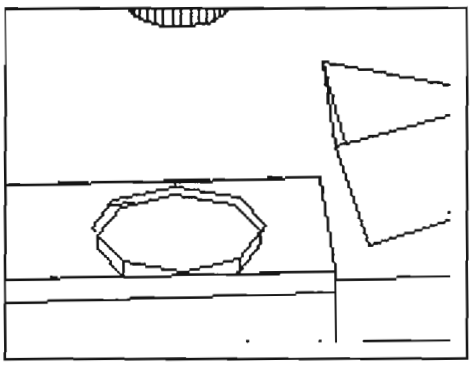
6.4 Laufzeitverhalten des Planers

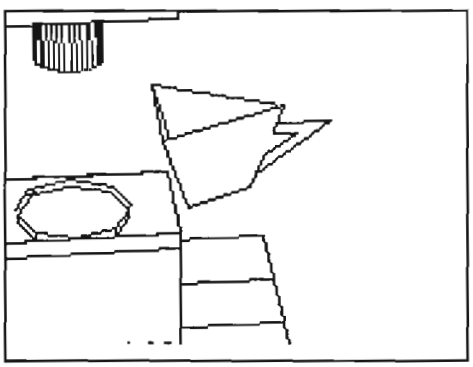
Der in BETTY verwandte Planungsansatz ist durch seine spezielle Konzeption zur Animationsplanung sehr effizient, da kein Suchraum aufgebaut wird, der mittels Backtracking abgesucht werden müßte. An dessen Stelle tritt die Entscheidung für eine von mehreren möglichen Dekompositionen in den bedingten Dekompositionsregeln. Das Laufzeitverhalten des Planers für den ungünstigsten Fall liegt bei

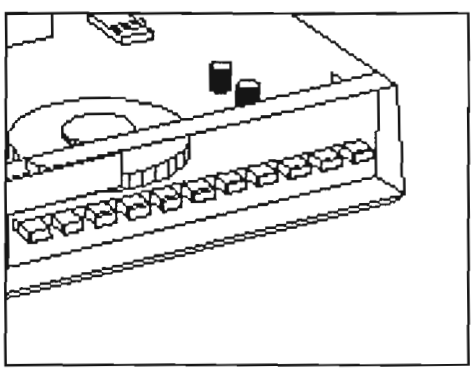
$$O(n \times m \times \log m)$$

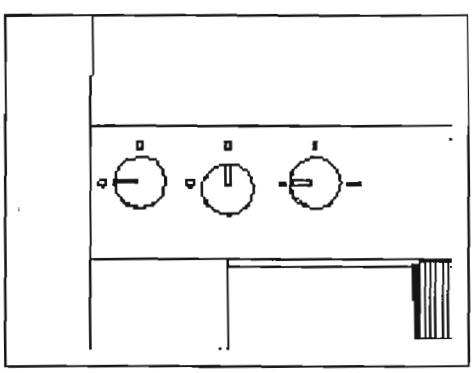
wobei n die Anzahl der vorhandenen Dekompositionsregeln ist, die schlimmstenfalls bei jeder Dekomposition alle abgesucht werden, und m die Anzahl der geplanten elementaren Aktionen. Mit einem besseren Algorithmus zur Suche der jeweiligen Dekompositionsregel (z.B. Anordnung in einem Graphen, in dem jeder











Knoten einem Regelkopf entspricht und fuer jedes mögliche Unterziel ein Verweis auf den entsprechenden Knoten existiert) ließe sich der Faktor n jedoch herauskürzen, so daß die Laufzeit dann nach einmaliger Erstellung des Graphen in quadratischer Zeit (über der Regelanzahl) nur noch $O(m \times \log m)$ beträgt.

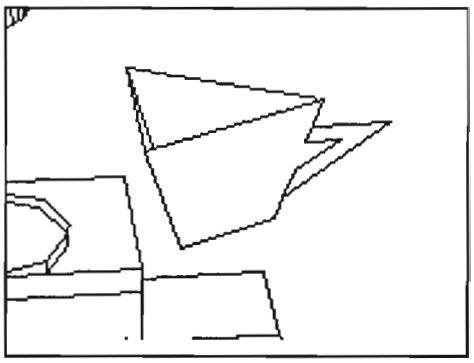
Eine Gefahr bilden die in Kap. 4.3.3 erwähnten rekursiven Regeln, falls sie so formuliert sind, daß eine unendliche Rekursion möglich wird. Diese Gefahr gilt es also bei der Formulierung der Regeln zu vermeiden, da eine automatische Überprüfung der Regeln auf diese Eigenschaft nicht möglich ist (Unentscheidbarkeit des Halteproblems).

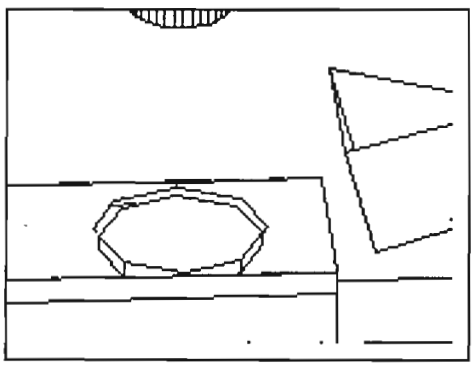
6.5 Demonstrationsumgebung des Systems

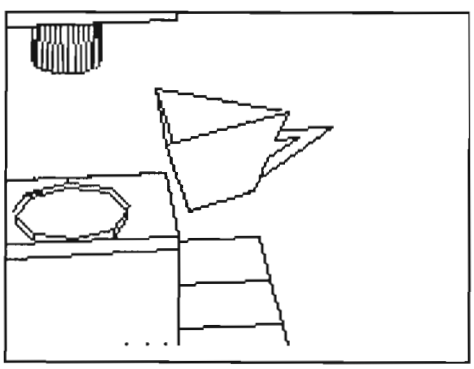
Zum Austesten und Demonstrieren der Komponente BETTY außerhalb des Gesamtsystems WIP steht eine Benutzeroberfläche zur Verfügung, die die interaktive Eingabe von Visualisierungszielen sowie das Planen, Berechnen und Abspielen der Animationen erlaubt (Bild 19). Die Visualisierungsziele werden in einem Menü ausgewählt, desgleichen die zugehörigen Parameter, Objekte, Zeitangaben usw..

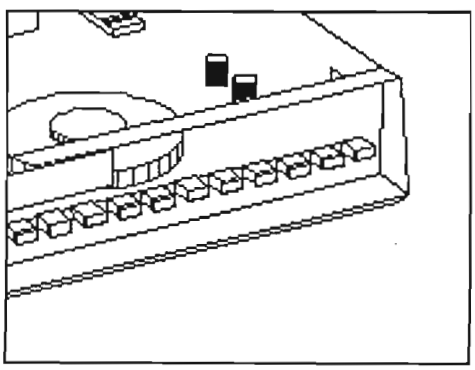
Die so erstellten Visualisierungsziele werden dem Planer übergeben, der ihre Dekomposition über die Benutzeroberfläche im oberen rechten Fenster in Form von Dekompositionsbäumen visualisiert. Außerdem werden die Berechnungen der Kamerapositionen und Einstellungen im oberen linken Fenster grafisch sichtbar gemacht und im Textfenster darunter die jeweiligen Veränderungen und Abfragen des Kontextes ausgegeben.

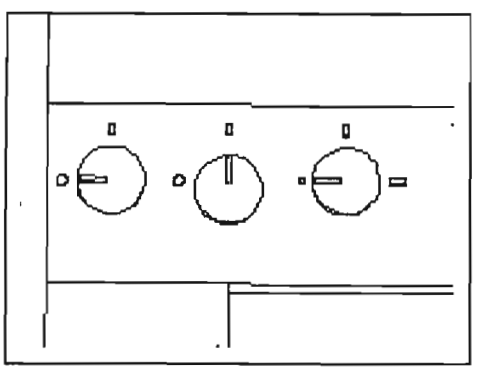
Diese Informationen ermöglichen ein genaues Verfolgen der Planung am Bildschirm und erleichtern das Erstellen und Modifizieren der Dekompositionsregeln dadurch sehr.











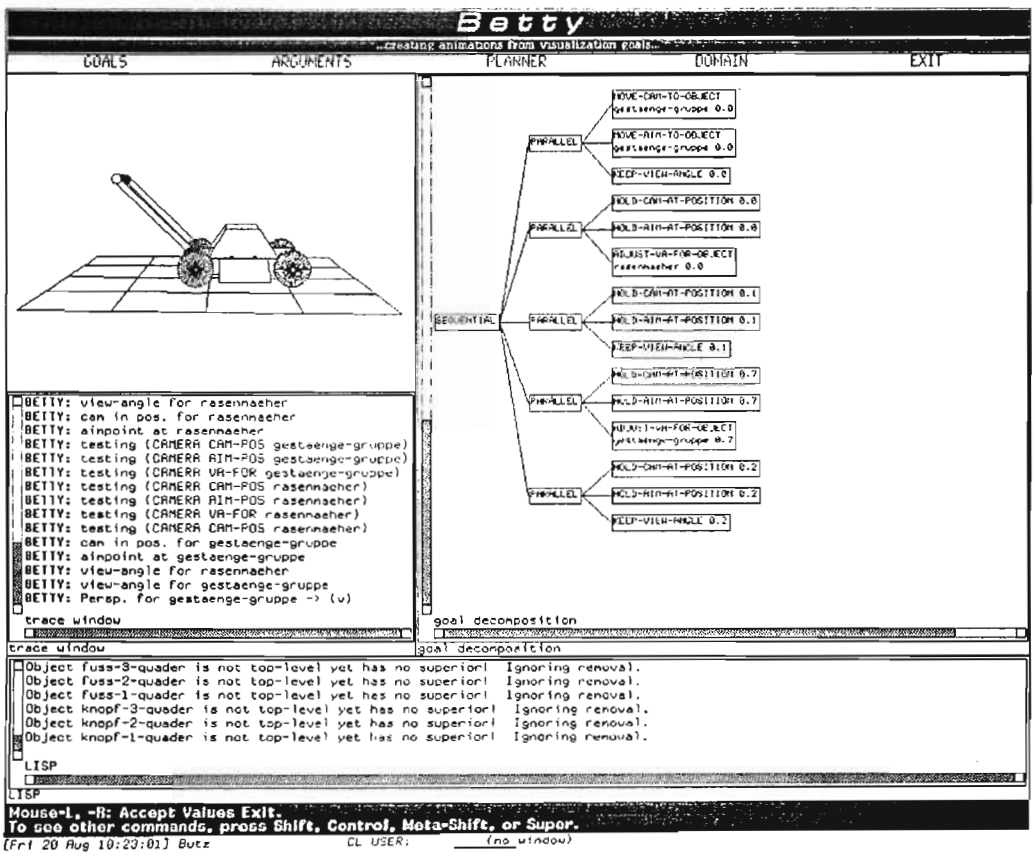
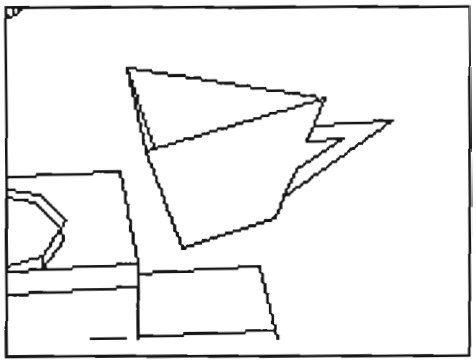
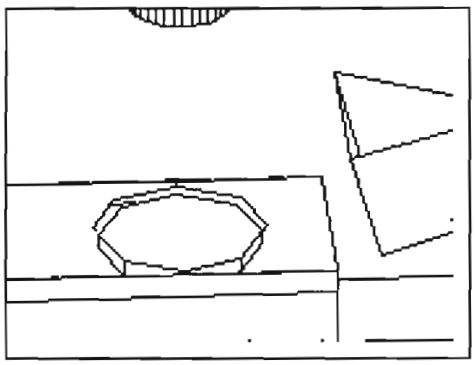
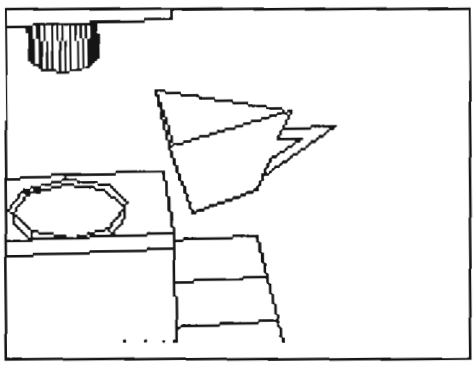
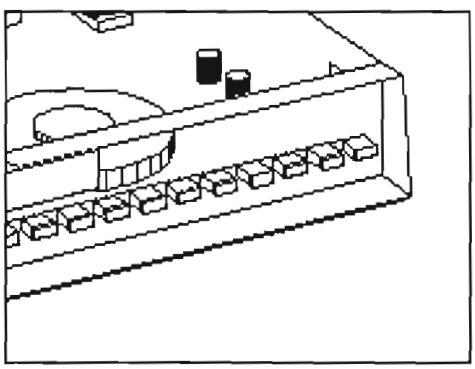


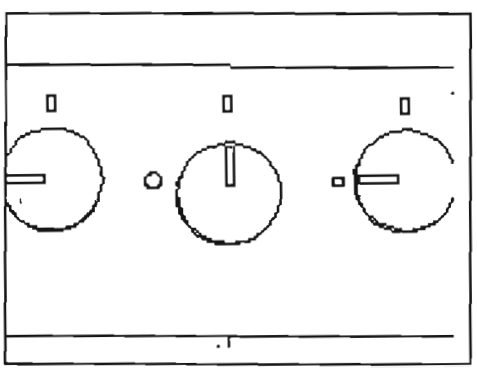
Abbildung 19: Demonstrationsumgebung zum BETTY-System











7 Vergleich mit anderen aktuellen Arbeiten

7.1 ESPLANADE (Feiner et al.)

Das derzeit an der Columbia University New York entwickelte System ESPLANADE (Expert System for PLANning Animation, Design, and Editing, beschrieben in [Ka93], Vorüberlegungen in [Ka90]) stellt ein wissensbasiertes System zur Planung von Animationssequenzen dar. Diesem System werden als Eingabe eine Folge von Aktionen übergeben, die zu visualisieren sind, sowie eine Menge kommunikativer Ziele. Diese kommunikativen Ziele beschreiben, welche Absicht mit der gesamten Animation verfolgt wird. Eine Animation wird in ESPLANADE, ähnlich wie in BETTY, hierarchisch geplant. Zunächst werden alle Sequenzen des Films festgelegt, dann alle Szenen jeder Sequenz und zuletzt alle Einstellungen jeder Szene.

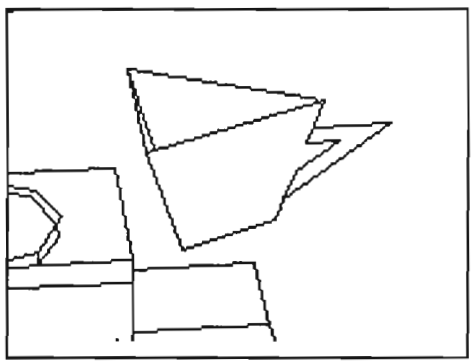
Hierarchie

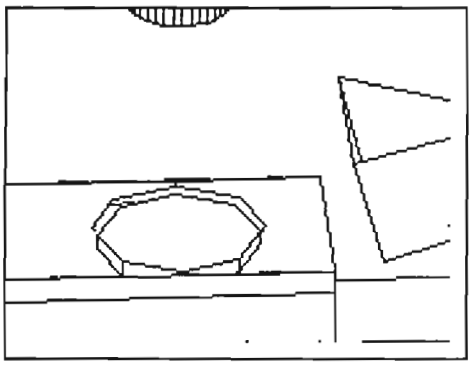
Die Autoren erläutern hierzu in [Ka93] ebenfalls, daß eine Animation eine hierarchische Struktur aufweist. Allerdings wird diese Struktur in ESPLANADE sehr eng mit der Struktur von Filmen verknüpft. Es findet eine direkte Zuordnung von Sequenzen, Szenen und Einstellungen, also Hierarchieebenen im Film zu Hierarchieebenen einer Animation statt. Die einzelnen Einstellungen der Animation werden dann in einer nachgeschalteten „inkrementellen Phase"des Planers „gedreht".

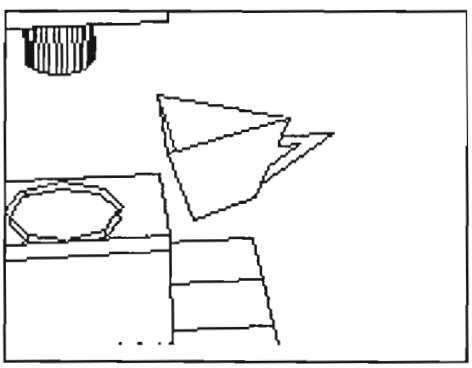
Diese Sichtweise ist zwar aus der Sicht eines Filmemachers motiviert und bietet auch eine nützliche Reihe von Analogien, sie läßt jedoch außer Acht, daß die Produktion eines Filmes und die Generierung einer Animation grundsätzlich verschieden ablaufen.

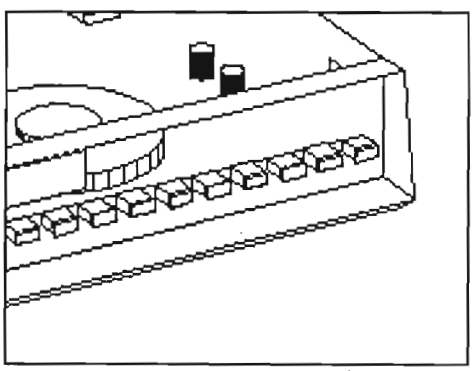
Beim Drehen eines Films werden zunächst alle Einstellungen gedreht, möglicherweise in mehreren Versionen, um sodann beim Schneiden zu Szenen, Sequenzen und zuletzt zum gesamten Film zusammengesetzt zu werden. So ergibt sich eine natürliche Untergliederung des Filmes durch Schnitte (bzw. Über- Aus- und Einblendungen als Sonderformen des Schnitts).

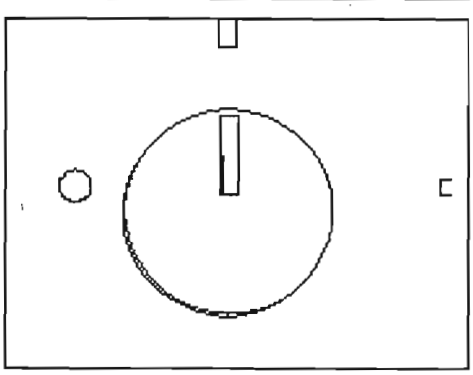
Bei der Erzeugung von Animationen finden jedoch das „Drehen"und „Schneiden"zur gleichen Zeit statt. Es werden keine Einstellungen mehrfach gedreht, da das System von vorneherein festlegt, welche Kamera- und Objektbewegungen ausgeführt werden sollen. Eine Nachbearbeitung findet nicht statt, so daß auch die natürliche Untergliederung durch Schnitte hinfällig wird. Stattdessen gibt es











eine Untergliederung in logische Einheiten, wie z.B. die Einteilung einer Einstellung in mehrere Phasen wie Ruhephase, Kamerafahrt und erneute Ruhephase oder die Unterteilung einer Bewegungsvisualisierung in die Lokalisation des zu bewegenden Objektes, die Bewegung selbst und eine anschließende statische Phase zur Festigung der neuen Situation.

Der hierarchische Aufbau einer Animation ist also einerseits oft sehr viel feiner strukturiert als der von [Ka93] angeführte vierstufige Aufbau eines Films, andererseits kann aber in einfachen Fällen auch eine Hierarchieebene der Animation mehrere Ebenen der Filmhierarchie überspringen. In jedem Fall muß überlegt werden, ob eine direkte Übertragung der Hierarchie adäquat ist.

Animationsbeschreibung

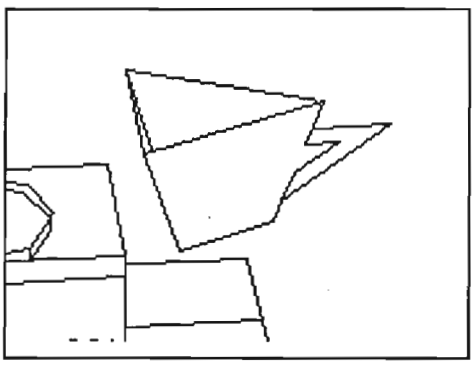
Die Zielsetzung des ESPLANADE-Systems geht insofern über die von BETTY hinaus, als nicht nur eine Animationssequenz mit einer festen Kamera und festem Layout generiert wird, sondern gleich eine komplette Spezifikation eines komplexen Animationslayouts. Die von der Hauptkamera aufgenommene Situation kann durch Insets mit dem Bild einer zweiten Kamera ergänzt werden, was zwar aus der konventionellen Filmtechnik weniger bekannt, dem neuen Medium jedoch vollständig angemessen ist.

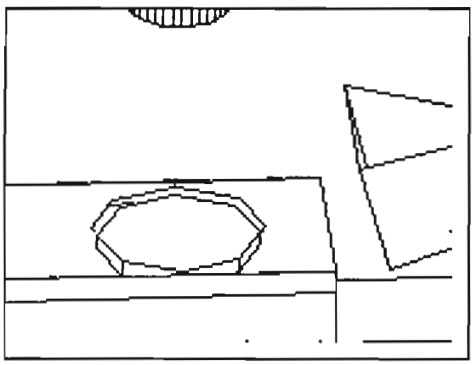
Dies hat allerdings zur Folge, daß der Planungsvorgang bis auf die Ebene einzelner Bilder fortgesetzt wird, so daß die von ESPLANADE erzeugten Animationen durch eine explizite Beschreibung jedes einzelnen Bildes spezifiziert sind. Der Planungsvorgang ist somit abhängig von der eigentlichen Animationsrealisierung und muß dies auch sein, da er sie ja wiederum bis ins Detail steuert. Diese an sich konsequente Fortsetzung des Hierarchie-Gedankens verhindert jedoch eine Trennung des Planungsvorganges von der Realisierung und legt beide aufeinander fest.

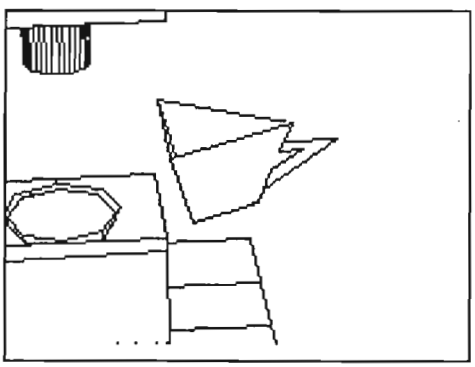
Planungsverfahren

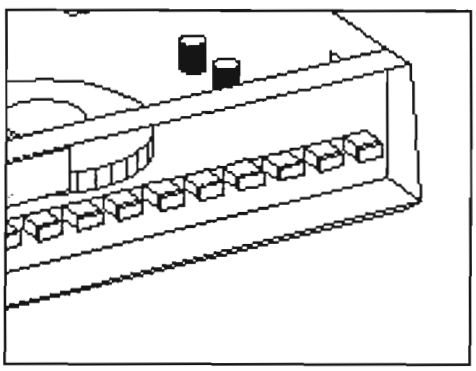
Obwohl sowohl BETTY als auch ESPLANADE hierarchische Planer zur Zergliederung der Animationsaufgabe einsetzen, besteht ein wesentlicher Unterschied in den Verfahren: Der in BETTY verwandte Planer arbeitet in die Tiefe des Animationsbaumes, während der ESPLANADE-Planer in die Breite arbeitet.

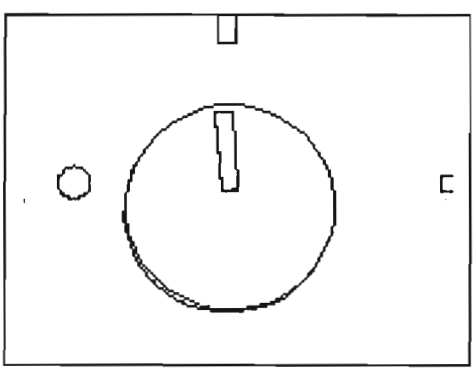
In ESPLANADE werden zuerst alle Sequenzen eines Films festgelegt, dann alle Szenen einer Sequenz, und schließlich alle Einstellungen einer Szene. Das hat











zur Folge, daß Details wie z.B. die Kameraposition aus vorangegangenen Einstellungen beim Planen einer neuen Szene nicht bekannt sind. Die nicht vorhandenen Angaben werden durch Heuristiken ersetzt.

BETTYs Planer hingegen dekomponiert jede Einstellung zuerst bis auf die Ebene der elementaren Aktionen und hat so zu jedem Zeitpunkt der Planung eine Repräsentation der schon abgelaufenen Aktionen, also des Zustandes der Szenerie zur Verfügung. Somit kann zu Beginn einer neuen Einstellung nicht nur nach heuristischen Gesichtspunkten, sondern anhand der aktuellen Situation entschieden werden, wie die nächste Einstellung abläuft.

Das System kann so beispielsweise aufgrund der räumlichen Situation entscheiden, ob es reicht, die Kamera nur etwas zu drehen oder ob ein Schnitt und ein Sprung zu einer völlig neuen Kameraposition nötig ist.

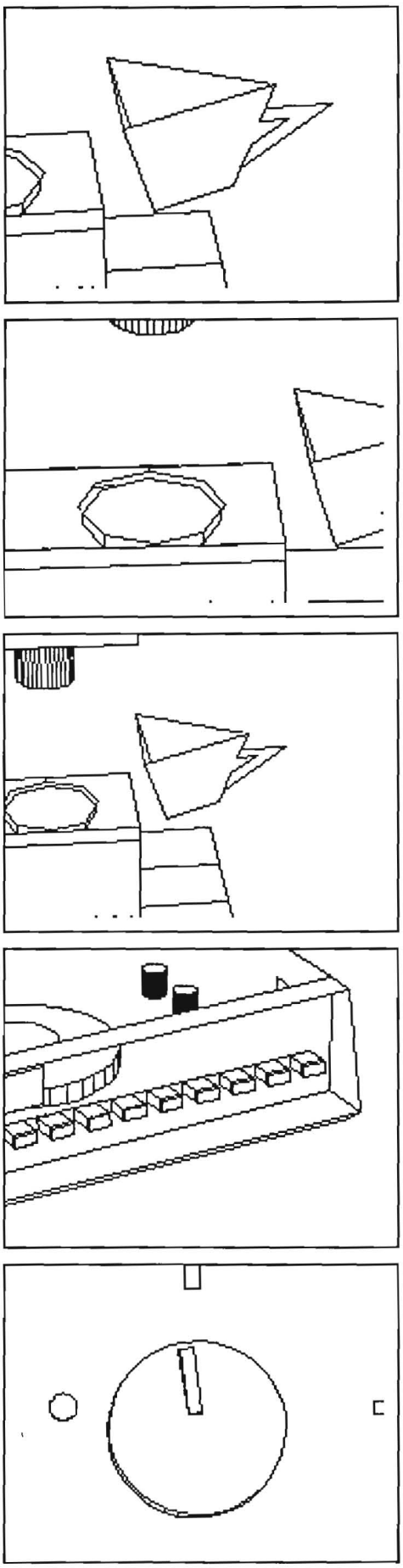
Andererseits erlaubt die Dekomposition in die Breite eine Voraussicht auf alle folgenden Filmteile der gleichen Hierarchieebene. Im Sinne konsistenter Präsentationen und eines flüssigen Ablaufes kann dies durchaus von Nutzen sein. Bei einer Dekomposition in die Tiefe ist eine solche Vorausschau natürlich nicht möglich. Welches Verfahren nun im Detail mehr Vorteile bietet, mag dahingestellt sein, daß beide aber gleichermaßen verwertbare Ergebnisse liefern, zeigen die Ergebnisse der beiden Systeme.

Inkrementalität

Schließlich unterscheiden sich BETTY und ESPLANADE auch in der inkrementellen Verwendbarkeit ihrer Planungskomponenten. Während in Esplanade der komplette Inhalt einer Animation zu Beginn des Planungsvorganges feststehen muß, damit eine Zergliederung in die Breite erfolgen kann, ist es in BETTY möglich, noch während des Planungsprozesses weitere Visualisierungsziele anzugeben, deren Realisierung dann mit der bisher geplanten Animation zu einem flüssigen zusammenhängenden Ganzen zusammengesetzt werden. Dies ist vor allem wichtig im Hinblick auf eine inkrementelle Verwendung der Systeme, in der sich Planungs- und Realisierungszeit überlappen. Ein Beispiel einer solchen Animation, die mehrere Visualisierungsziele realisiert, bildet Daumenkino Nr. 5, in dem nacheinander drei verschiedene Bewegungen gezeigt werden.

7.2 AnimNL (Badler et al.)

Norman Badler und Bonnie Webber stellen in [Bad90, We92, BPW93] ein System vor, das Animationen aus natürlicher Sprache generiert. Von der Annahme aus-



gehend, daß es für Handlungen, gleich ob sie in natürlicher Sprache oder durch Animationen beschrieben werden, eine gemeinsame Repräsentation geben muß, extrahiert dieses System zuerst aus einer Eingabe in natürlicher (englischer) Sprache eine solche Repräsentation, um aus ihr dann eine Animation zu generieren.

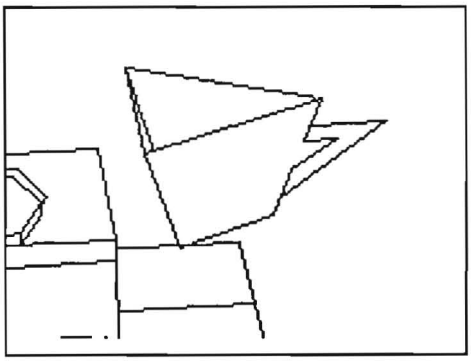
Animationsbeschreibung

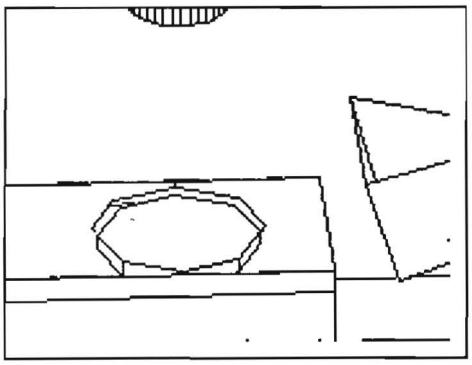
Die Beschreibung der erzeugten Animation erfolgt dabei auf Ebene eines virtuellen Agenten namens Jack, der sich in einer virtuellen Welt bewegt und die im Animationsplan generierten Anweisungen ausführt. Solche Anweisungen sind grundlegende Befehle wie „bewege Objekt X nach Punkt Y", „Gehe nach Z", „Greife Objext X"und ähnliches. Die Kameraführung, die das Zeigen der Animation ja erst ermöglicht, wird dabei nach der vorliegenden Literatur ([Ba90]) auf einfache Defaultpositionen zurückgeführt und laut [We92] durch einen menschlichen Animator ausgeführt. Sie spielt im AnimNL System eher eine Nebenrolle, wohingegen sie zentraler Bestandteil der von BETTY erzeugten Animationsscripte ist.

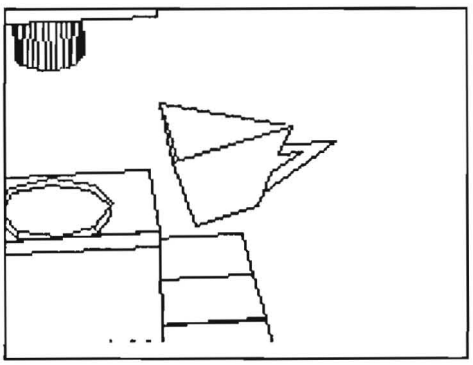
Der Schwerpunkt des AnimNL Systems liegt vielmehr auf der exakten Extraktion der Aktionen des Agenten aus den Anweisungen in natürlicher Sprache. Hier werden sehr weitreichende Untersuchungen bezüglich der Begriffe Handlung und Plan angeführt. Auch die Steuerung des Agenten selbst ist in [BPW93] eingehend untersucht. Außerdem werden die von AnimNL erzeugten Animationen als „erzählte"Animationen (narrated animations) präsentiert, also mit Unterstützung durch gesprochene Sprache. Hierdurch ist es (wie auch in dieser Arbeit an anderer Stelle gesagt) möglich, kausale Zusammenhänge zu visualisieren, die durch Animation alleine so nicht darstellbar wären.

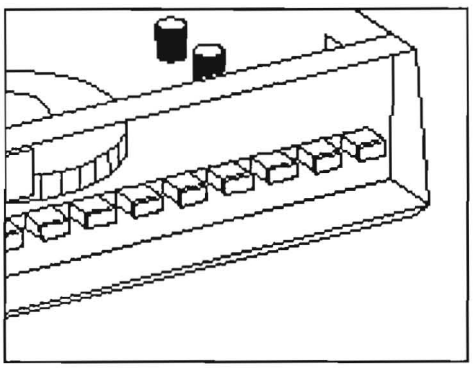
Repräsentation

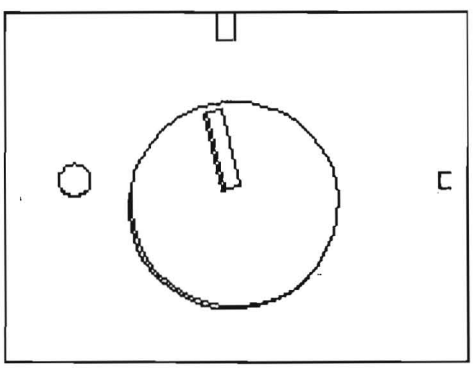
Letzten Endes bleibt die zentrale These, daß Sprache und Animation verschiedene Beschreibungen der gleichen Aktionen sind und daher auch von einer Beschreibung in die andere überführt werden können bzw. aus einer gemeinsamen Repräsentation der Handlungen erzeugbar sein müssen. Einer etwas schwächeren Version dieser Forderung entsprechen auch die Systeme BETTY und TOPAS, die aus der gleichen Repräsentation von Bewegungen (siehe 4.7) zwei verschiedene Präsentationen erzeugen, nämlich Animation und (unbewegte) technische Grafik.











8 Die nächsten Schritte

Die bis zu diesem Punkt realisierten Verfahren ermöglichen zwar bereits einen eindrucksvollen Leistungsumfang des Systems, jedoch sind Erweiterungen und Abwandlungen in verschiedene Richtungen denkbar. Als nächster Schritt wäre beispielsweise der Austausch der Realisierungskomponente interessant, um eine optisch ansprechendere Ausgabe BETTYs zu erreichen.

8.1 Einbeziehung des Lichts in den Planungsprozeß

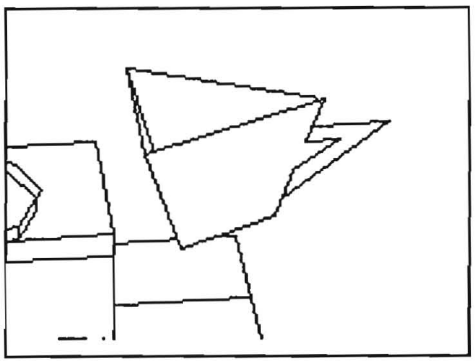
Die meisten Animationssysteme auf leistungsfähigeren Rechnern ermöglichen eine Bilderzeugung mit Hilfe des Raytracing-Verfahrens oder anderer mathematischer Modelle (siehe [Fo90]), die eine Darstellung von Licht und Schatten im Bild erlauben. S-Dynamics bzw. S-Geometry bieten diese Möglichkeit zwar prinzipiell auch, jedoch sind die Ergebnisse aus verschiedenen Gründen im Moment wenig ansprechend: Das Lichtmodell in S-Geometry weicht beispielsweise grob von der Natur ab, indem die Beleuchtungsstärke nicht wie in Wirklichkeit mit dem Quadrat der Entfernung von der Lichtquelle abnimmt, sondern unabhängig vom Abstand gleich bleibt.

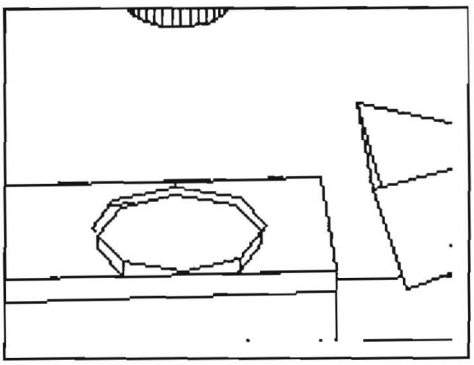
Außerdem besteht mit der vorhandenen Hard- und Software nicht die Möglichkeit, derart „gerenderte"(d.h. mit Licht und Schatten dargestellte) Bilder vernünftig auszugeben. Durch einen Fehler in der Bildverarbeitung unter Genera werden sogar die Tonwerte beim Ausdruck oder der Ausgabe in eine Datei invertiert: Die Bildfolgen 16 bis 18 entstanden jeweils mit explizit vorgegebenen Beleuchtungen und erreichen so trotz invertierter Darstellung einigermaßen überzeugende Tonwertverteilungen.

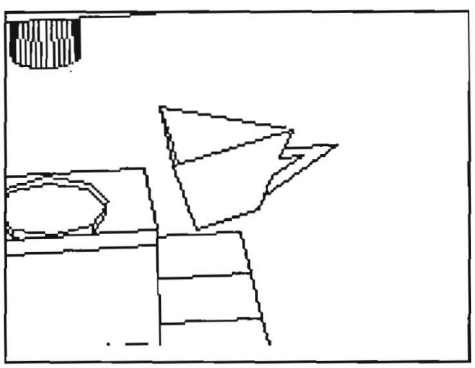
Setzt man also eine andere Realisierungskomponente voraus, die die erwünschten Ergebnisse liefert, so stellt sich bei der Verwendung von Licht und Schatten sofort ein neues Problem:

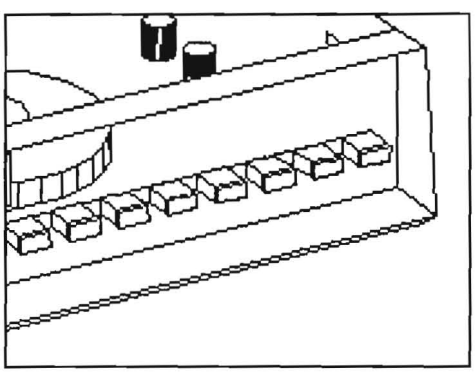
8.1.1 Lichtführung

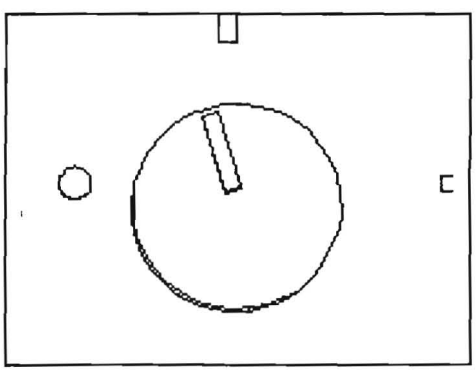
Wie sollen während einer Animation die Beleuchtungselemente gesetzt werden? Welche Beleuchtung ist nötig, um ein Objekt mit all seinen Details gut erkennbar zu machen? Wie kann man verhindern, daß (wie in schlechten Filmen) Elemente dieser Hilfsinstallation (Lampen, Aufheller, Abdunkler, Reflektoren) im Bild erscheinen? Durch welche Beleuchtungsformen können Bildaussagen unterstrichen











oder verstärkt werden, welche Beleuchtungen sind der Vermittlung der Information zuträglich?

Als Beispiel wäre die Lokalisierung eines Teiles in einem System mit Licht und Schatten nicht nur durch entsprechende Kameraführung sondern auch durch durch gezielte Beleuchtung, beispielsweise durch einen Spot auf das Teil und eine dunklere Darstellung der Umgebung möglich. Weitere solche Möglichkeiten sind in [Du93] vorgeschlagen und versprechen eine erhebliche Bereicherung der Ausdrucksmittel.

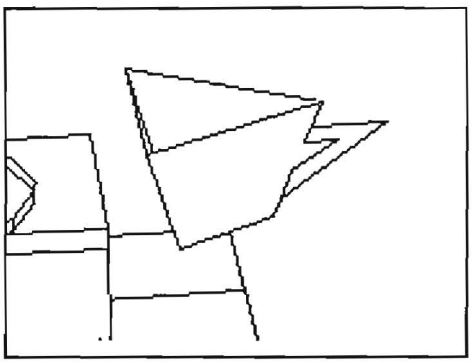
Durch ein Erweitern des Regelwerkes auf explizite Bewegungen und Positionen von modellierten Lampen wäre wohl eine erste Lösung erreichbar, um jedoch alle Möglichkeiten einer solchen Erweiterung der Ausgabemöglichkeiten auszuschöpfen, ist wohl eine prinzipielle Erweiterung des Ansatzes vonnöten. Ähnlich der Perspektivwahl zur Bestimmung von Kamerapositionen erfordert die Beleuchtung einer Szene eine umfangreiche „Lichtwahlkomponente"zur Bestimmung der Positionen von Lampen, Aufhellern und Reflektoren zur Erreichung der gewünschten Gesamtwirkung und gleichzeitig ohne die geforderten Nebenbedingungen zu verletzen.

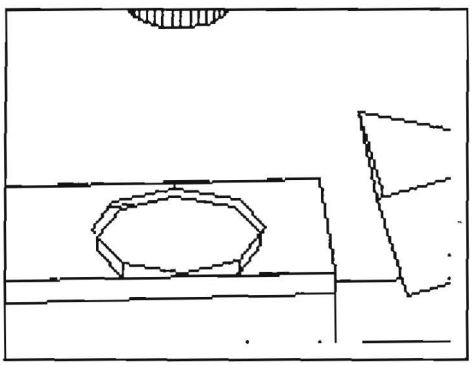
8.1.2 Einbeziehung von Farbe

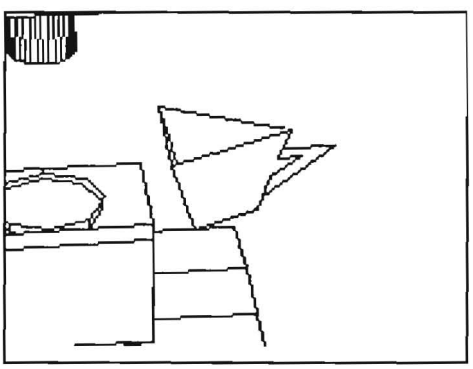
Während die Erweiterung um Licht und Schatten also eine konzeptionelle Erweiterung wäre, läßt sich die Dimension Farbe direkt in das bestehende Konzept integrieren. Farbige Objekte werden als solche modelliert, farbliche Veränderungen sind konzeptionell nicht verschieden von geometrischen Veränderungen (d.h. Bewegungen). Es müßte also lediglich ein weiterer Bewegungstyp neben Rotation, Translation und Trajektorienbewegung eingeführt werden, nämlich die Farbveränderung. Diese Erweiterung eröffnet aber wiederum neue Möglichkeiten der Darstellung, wie wir gleich sehen werden.

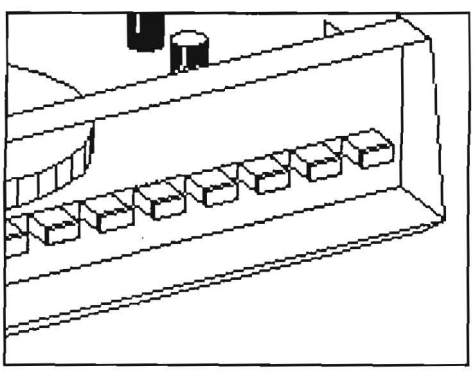
8.2 Metagrafik in Animationen

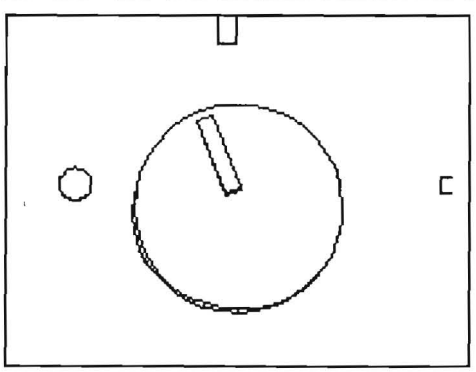
Eine denkbare konzeptuelle Erweiterung des BETTY Systems ist die Einführung metagrafischer Elemente. Solche Elemente können sich einerseits auf Veränderungen der vorhandenen Objekte beziehen, andererseits aber auch die Einführung von Metaobjekten bedeuten.











8.2.1 Vorgänge ohne geometrische Äußerungsform

Wie ließe sich beispielsweise die Größe „Temperatur"in einer Animation darstellen? In statischen Grafiken wird herkömmlicherweise auf eine Darstellung durch ein metagrafisches Thermometer zurückgegriffen, zuweilen auch die Analogie zu den Farben Rot und Blau, wie wir sie vom Wasserhahn her kennen, benutzt.

Diese Visualisierungen einer Größe ohne geometrische Äußerungsform lassen sich auch in Animationen einsetzen: Ein sich erwärmendes Teil könnte sich (Licht und Farbe in der Ausgabekomponente vorausgesetzt) rot verfärben, ein abkühlendes Objekt blau. Zusätzlich ist ein metagrafisches Thermometer denkbar, das die Farbveränderung semantisch eindeutig macht, indem z.B. seine Quecksilbersäule steigt oder fällt.

8.2.2 Unterstützung sichtbarer Vorgänge

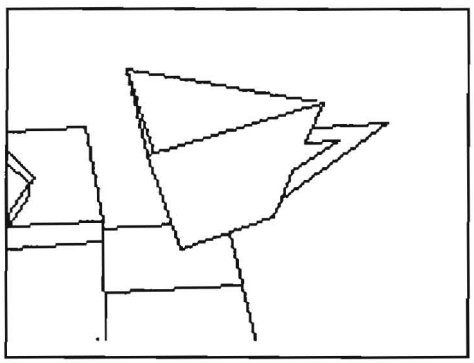
Zur Zeit bewegen sich die Objekte in der Modellwelt BETTYs wie von Geisterhand, nämlich ohne ersichtlichen Grund. Soll beispielsweise eine Klappe geöffnet werden, so bewegt sie sich in der Animation zwar auf eine entsprechende Art und Weise, jedoch ist nicht ersichtlich, ob die Bewegung durch einen Menschen hervorgerufen werden muß, der diese Klappe bedient, oder ob es vielleicht einen versteckten Motor gibt, der die Bewegung hervorruft.

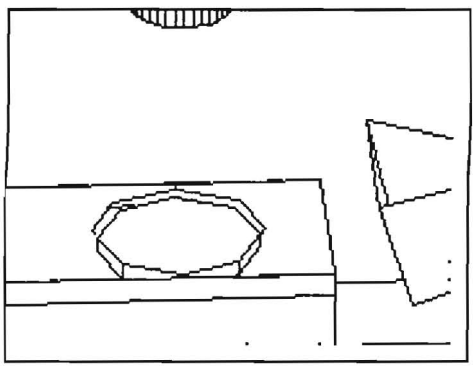
Dieses konzeptuelle Problem wäre lösbar durch einen virtuellen Agenten in der Modellwelt, ähnlich dem in AnimNL verfolgten Ansatz. Ob es sich dabei um eine komplett modellierte Person, um eine Hand oder lediglich einen Finger handelt, hängt wiederum von der Situation ab und muß je nach der erwünschten Bildaussage entschieden werden. Als Anregung zum Weiterspinnen dieses Gedankens und aller seiner Folgen betrachte man die Rolle und Möglichkeiten des eiskalten Händchens in den Filmen der Adams Family.

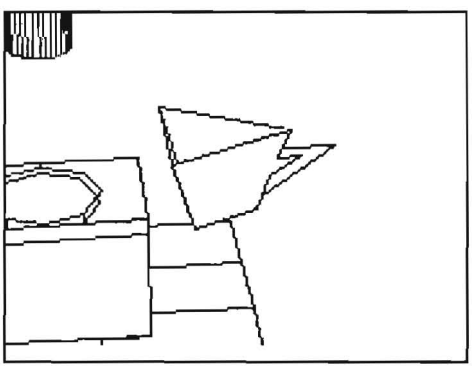
8.3 Abstraktion in Animationen

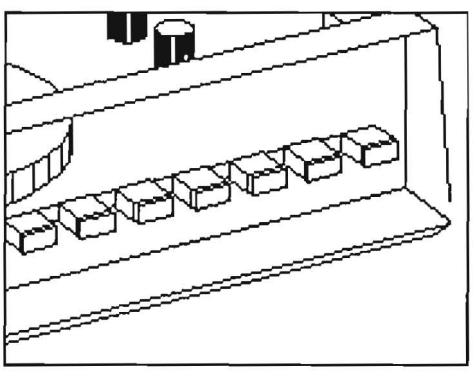
In statischen Grafiken ist es außerdem möglich, durch gezielte Abstraktion die Bildaussage zu unterstützen, beispielsweise indem alle bildunwichtigen Details bis auf bloße Sichtbartkeit abstrahiert, die wichtigen Details aber verstärkt werden. In WIP wird dies bei der Generierung statischer Grafiken durch das System PROXIMA realisiert. Details hierzu sind in [Kr94] erläutert.

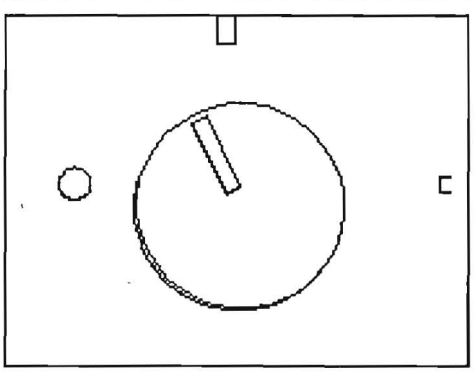
Neben der Unterstützung der Bildaussage bietet die Abstraktion geometrischer Modelle bei ihrer Animation aber noch einen weiteren wesentlichen Vorteil:



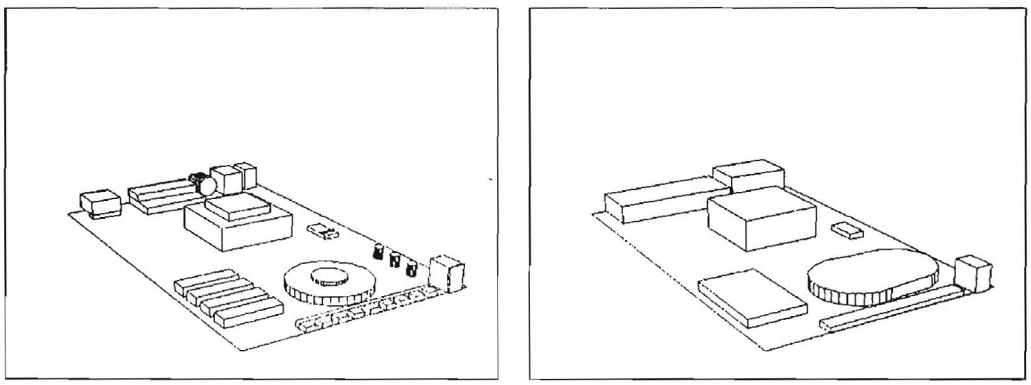








Durch die Reduktion der geometrischen Komplexität der Modelle, das Verschmelzen mehrerer Objekte zu einem oder das Verschwinden mancher Objekte sinkt die Datenmenge, die bei der Berechnung der Animation bewältigt werden muß, erheblich.

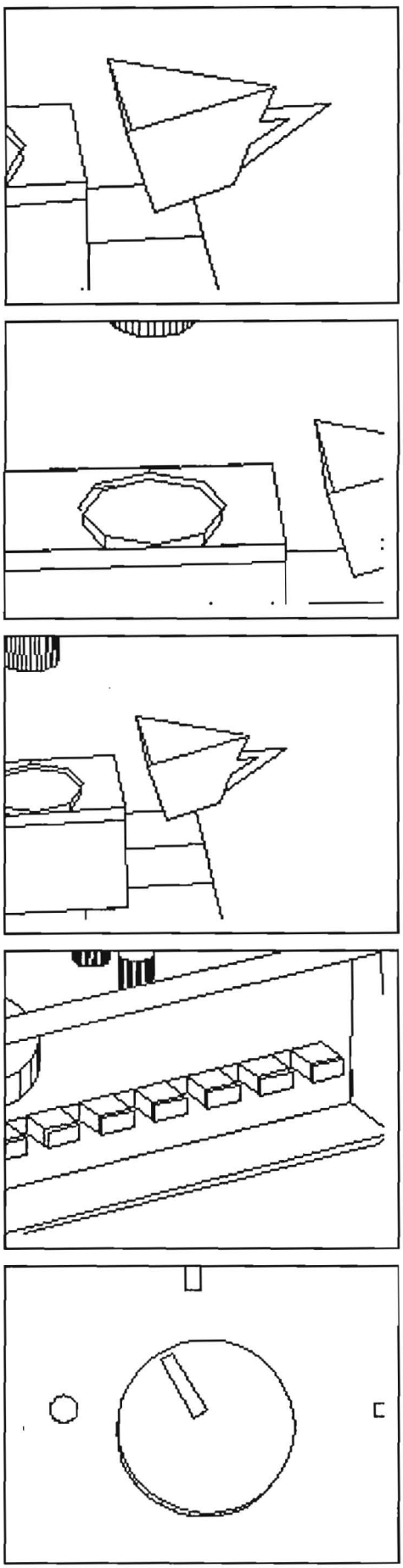


Original: 38 Objekte Abstraktion: 9 Objekte

Abbildung 20: Datenreduktion durch Abstraktion

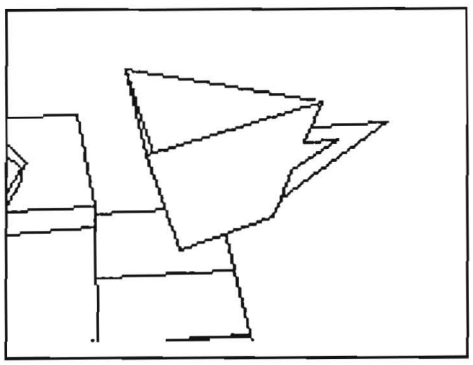
Eine Animation, die lediglich abstrahierte Modelle derjenigen Objekte verwendet, die nicht zentraler Bestandteil der Information sind, läßt sich in wesentlich kürzerer Zeit berechnen und begünstigt daher Echtzeitanwendungen des Systems enorm. Der Rechenaufwand zur Vereinfachung der Modelle kommt zwar hinzu, steht aber bei der derzeitigen Realisierung des PROXIMA Systems in einem sehr guten Verhältnis zur Zeitersparnis bei der Berechnung jedes einzelnen Bildes.

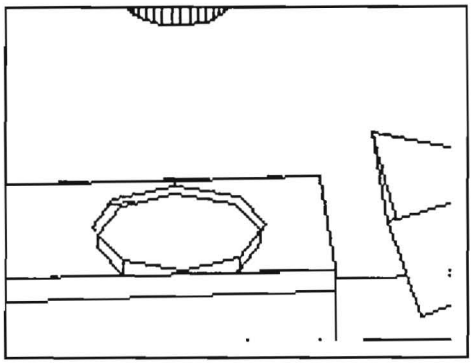
In jedem Falle gibt es eine Reihe vielversprechender Erweiterungsmöglichkeiten des Systems BETTY , die jedoch sämtlich den Umfang dieser Diplomarbeit übersteigen würden und deshalb als Anregung für künftige Arbeiten auf dem Gebiet der automatischen Animationsgenerierung gedacht sind.

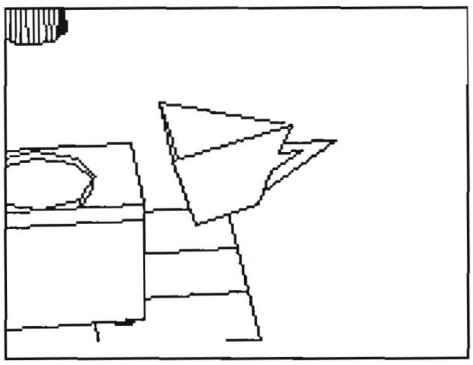


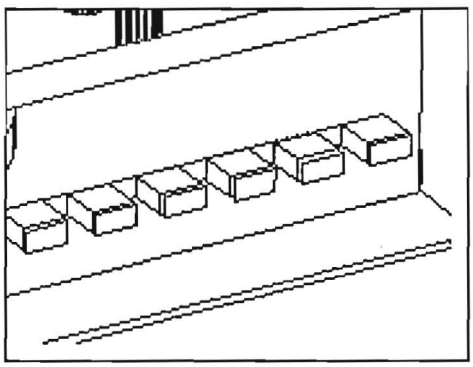
A Auflistung der Dekompositionsregeln

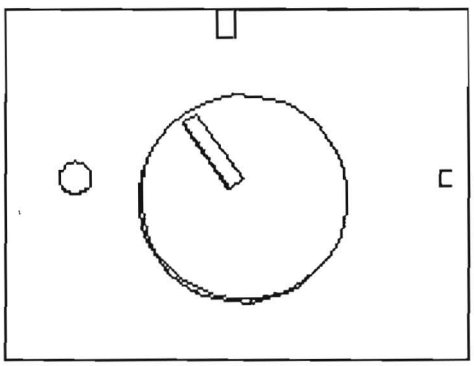
```
;;; -*- Mode: Lisp; Syntax: Common-Lisp; Package: dyna; Base: 10;
       Default-character-style: (:FIX :ROMAN :NORMAL); -*-

;;; Dekompositionsregeln fuer die Umsetzung von Vis.zielen in elem. Aktionen:
;;; Elem. Aktionen sind:
;;; move-cam-to-object   (part time) - Kamerapos. fahren, dass Objekt sichtbar
;;; hold-cam-at-position (time)      - Kameraposition festhalten
;;; move-aim-to-object   (part time) - Zielpunkt auf Objektmitte fahren
;;; hold-aim-at-object   (part time) - Zielpunkt auf Objektmitte halten
;;; hold-aim-at-position (time)      - Zielpunkt festhalten
;;; adjust-va-for-object (part time) - Brennw. fahren, dass Objekt bildfuellend
;;; keep-view-angle      (time)      - Brennweite festhalten
;;; move-object          (part motion start end time) - Obj. entlg einer Bew.
;;; move-object-and-aim  (part motion start end time) - Obj. und Zielp. entlg
;;;                                                     einer Trajekt. bewegen
;;; explode-object       (part time) - Objekt explodieren (alle Teile parallel)

;;; Kamera-Grundbausteine

(anim-rule hold-view              (time)        ; alles fest lassen
           :parallel
           (hold-cam-at-position  time)
           (hold-aim-at-position  time)
           (keep-view-angle       time))

(anim-rule follow-aim             (part time) ; Zielpunkt folgt dem Objekt
           :parallel
           (hold-cam-at-position  time)
           (hold-aim-at-object    part time)
           (keep-view-angle       time))

(anim-rule follow-aim-and-zoom    (part time) ; Zielpunkt und Brennweite
           :parallel                          ; folgen dem Objekt
           (hold-cam-at-position  time)
           (hold-aim-at-object    part time)
           (adjust-va-for-object  part time))
```



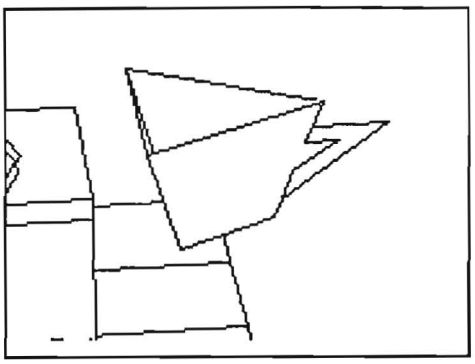


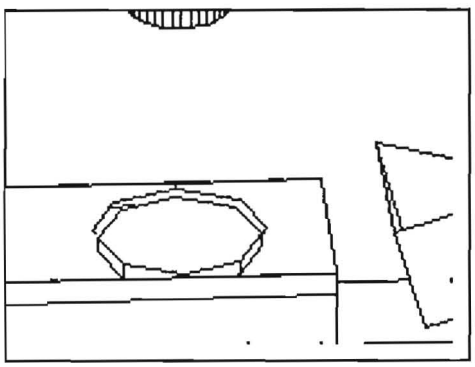


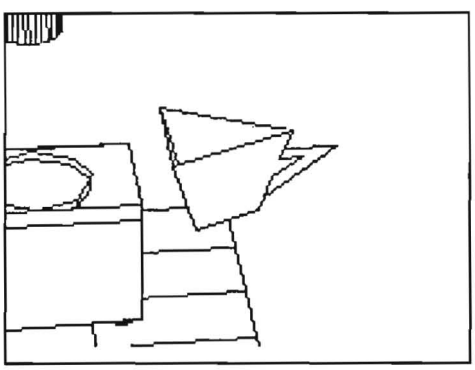


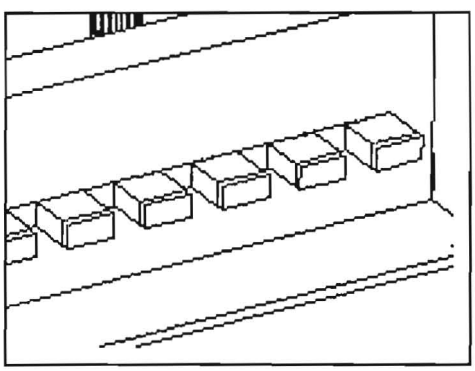


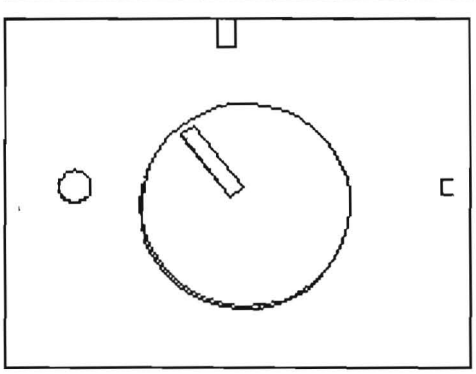
```
(anim-rule zoom-to-object          (part time) ; Brennweite faehrt aufs Objekt
           :parallel
           (hold-cam-at-position  time)
           (hold-aim-at-position  time)
           (adjust-va-for-object  part time))

(anim-rule go-aim-to-object        (part time) ; Zielpunkt geht aufs Objekt
           :parallel
           (hold-cam-at-position  time)
           (move-aim-to-object    part time)
           (keep-view-angle       time))

(anim-rule zoom-aim-to-object      (part time) ; Zielpunkt und Brennweite
           :parallel                           ; fahren aufs Objekt
           (hold-cam-at-position  time)
           (move-aim-to-object    part time)
           (adjust-va-for-object  part time))

(anim-rule move-view-to-object     (part time) ; Kamerapos. und Zielpunkt
           :parallel                           ; fahren aufs Objekt
           (move-cam-to-object    part time)
           (move-aim-to-object    part time)
           (keep-view-angle       time))

(anim-rule zoom-view-to-object     (part time) ; Pos., Ziel und Brennweite
           :parallel                           ; fahren aufs Objekt
           (move-cam-to-object    part time)
           (move-aim-to-object    part time)
           (adjust-va-for-object  part time))

(anim-rule cut-view-to-object      (part time) ; Schnitt auf Kamerapos.
           :sequential                         ; und aim fuer Objekt
           (move-view-to-object   part 0)
           (hold-view             time))

(anim-rule cut-all-to-object      (part time)  ; Schnitt auf Kamerapos,
           :sequential                         ; v-a und aim fuer Objekt
           (zoom-view-to-object   part 0)
           (hold-view             time))
```



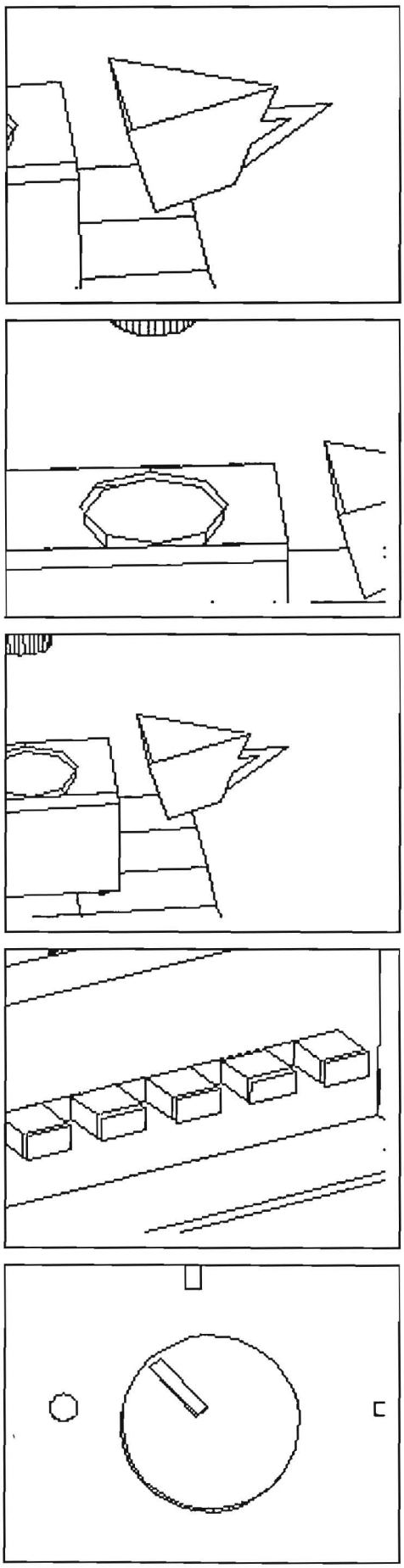




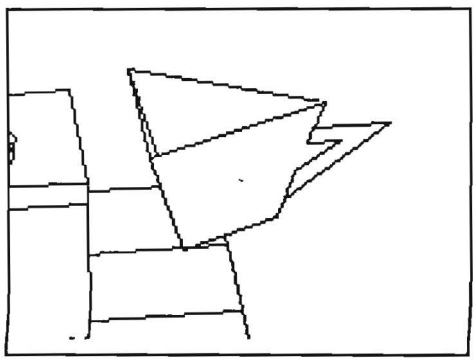


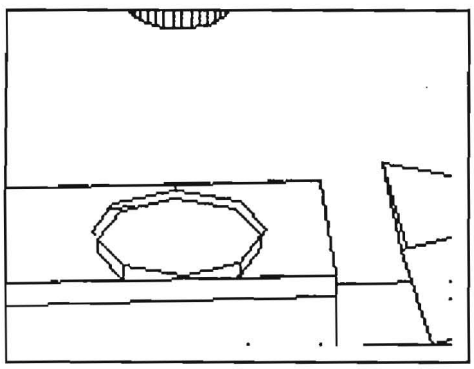


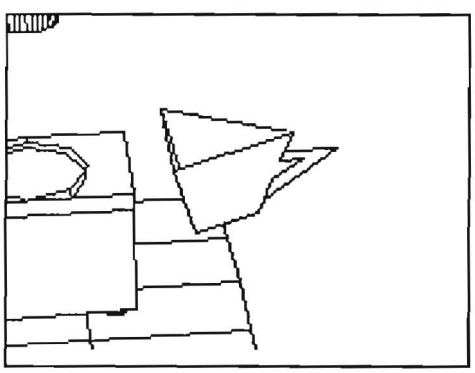
```
;;; Bewegungs-Grundbausteine

(anim-rule show-object-move       (part motion start end time)
          :parallel               ; Objekt bewegen ,Kamera fest
          (move-object            part motion start end time)
          (hold-view              time))

(anim-rule goto-object-move       (part motion start end time)
          :parallel               ; Objekt bewegen ,Kamera geht aufs Objekt
          (move-object            part motion start end time)
          (go-aim-to-object       part time))

(anim-rule follow-object-move     (part motion start end time)
          :parallel               ; Objekt bewegen und Ziel mitfahren
          (hold-cam-at-position   time)
          (keep-view-angle        time)
          (move-object-and-aim    part motion start end time))

(anim-rule mixed-show-object-move (part motion time) ; Objekt bewegen und Ziel
          :sequential                                ; in der 2. Haelfte mitfahren
          (show-object-move           part motion 0.0 0.4 (* time 0.4))
          (goto-object-move           part motion 0.4 0.7 (* time 0.3))
          (follow-object-move         part motion 0.7 1.0 (* time 0.3)))

(anim-rule show-object-explode    (part time) ; Objekt explodieren ,Kamera fest
          :parallel
          (explode-object         part time)
          (hold-view              time))

;;; Bildeinstellungen, die das Objekt waehrend der ganzen Bewegung
;;; sichtbar lassen:

(anim-rule zoom-view-to-object-and-motion (part motion time)
          :parallel                              ; Pos., Ziel und Brennweite
                                                 ; fahren aufs Objekt und
          (move-cam-to-object     part time)     ; seine Trajektorie, aimpoint
          (move-aim-to-object     part time)     ; ist Objektmitte
          (adjust-va-for-object   (part motion) time))
```

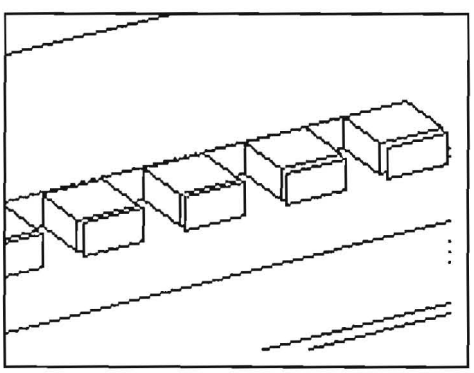


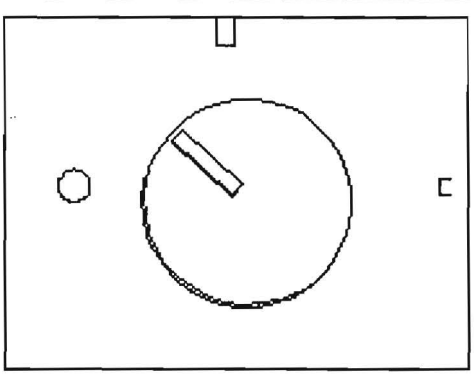
```
(anim-rule zoom-view-to-motion-and-object (part motion time)
          :parallel                              ; Pos., Ziel und Brennweite
                                                 ; fahren aufs Objekt und
          (move-cam-to-object    part time)      ; seine Trajektorie, aimpoint
          (move-aim-to-object    motion time)    ; ist Mitte der Trajektorie
          (adjust-va-for-object  (part motion) time))

(anim-rule cut-view-to-object-and-motion    (part motion time)
          :sequential                           ; Pos., Ziel und Brennweite
                                                ; schneiden aufs Objekt und
          (zoom-view-to-object-and-motion  part motion 0) ; seine Trajektorie,
          (hold-view                       time)) ; aimpoint ist Objektmitte

(anim-rule cut-view-to-motion-and-object    (part motion time)
          :sequential                           ; Pos., Ziel und Brennweite
                                                ; schneiden aufs Objekt und
          (zoom-view-to-motion-and-object  part motion 0) ; seine Trajektorie,
                                                ;  aimpoint ist Objektmitte
          (hold-view                       time))

;;; Auf den Grundbausteinen aufbauende Operationen:
;;; Achtung: keine 2 Grundbausteine parallel !
;;; und keine elem. Aktionen parallel zu Grundbausteinen !

(anim-rule show-object           (part time)    ; Objekt ins Bild mit Schwenk
          :sequential
          (move-view-to-object   part (* time 0.8))
          (hold-view             (* time 0.2)))

(anim-rule show-object-near-1    (part time)    ; Objekt bilfuellend ins Bild
          :sequential                           ; nacheinander Kamera und
                                                ; aimpoint bewegen
          (move-view-to-object   part (* time 0.4))
          (zoom-to-object        part (* time 0.4))
          (hold-view             (* time 0.2)))

(anim-rule show-object-near-2    (part time)    ; Objekt bilfuellend ins Bild
          :sequential                           ; parallel Kamera und
                                                ; aimpoint bewegen
          (zoom-view-to-object   part (* time 0.8))
          (hold-view             (* time 0.2)))
```



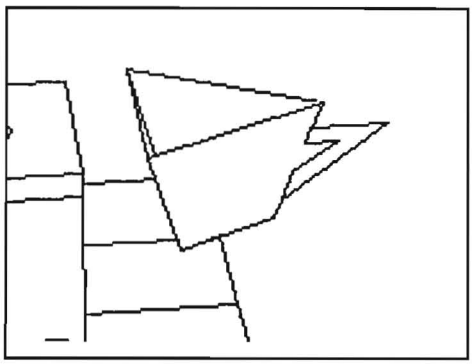


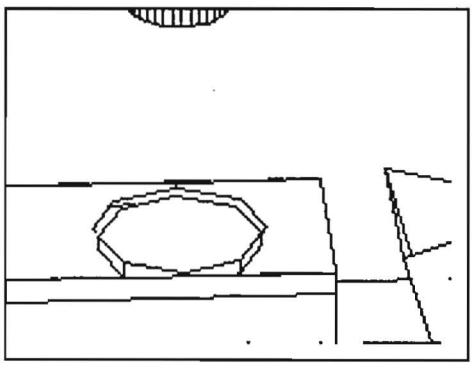


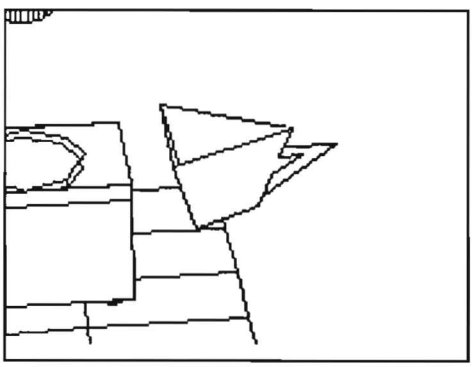


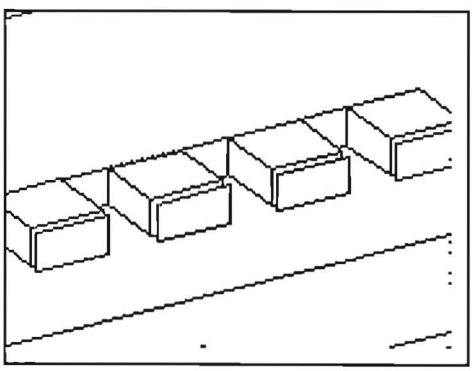


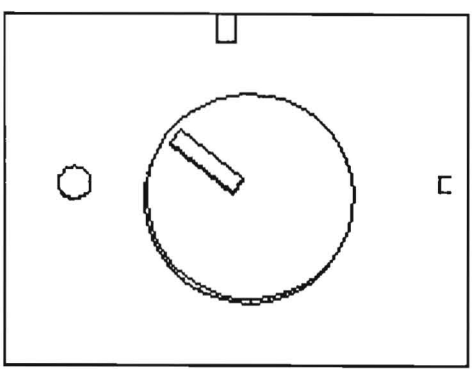
```
;;; Lokalisation und Bewegungen in Bezug zu anderen Objekten
;;; (irt = in relation to)

(anim-rule localize-object-irt    (part whole time) ; Objekt in Relation zu
          :sequential                               ; einem anderen lokalis.
          (zoom-aim-to-object     whole (* time 0.2))
          (move-view-to-object    part (* time 0.3))
          (zoom-to-object         part (* time 0.3))
          (hold-view              (* time 0.2)))

(anim-rule show-motion-irt        (part whole motion time) ; weg und zum Objekt
                                                           ; fahren und
          :sequential                                      ; Bewegung mit fester
          (zoom-aim-to-object     whole (* time 0.1))      ; Kamera zeigen
          (move-view-to-object    whole (* time 0.1))
          (zoom-view-to-motion-and-object part motion (* time 0.1))
          (show-object-move       part motion 0.0 1.0  (* time 0.7)))

(anim-rule show-motion-following-irt (part whole motion time) ; weg und zum Objekt
                                                              ; fahren und
          :sequential                                         ; Bewegung zeigen,
          (zoom-aim-to-object     whole (* time 0.1))         ; Zielp. faehrt mit
          (move-view-to-object    part (* time 0.1))
          (zoom-view-to-object-and-motion part motion (* time 0.1))
          (follow-object-move     part motion 0.0 1.0 (* time 0.7)))

(anim-rule show-motion-mixed-irt  (part whole motion time) ; weg und zum Objekt
                                                           ; fahren und
          :sequential                                      ; Bewegung zeigen,
          (zoom-aim-to-object     whole (* time 0.1))      ; Zielpunkt in der
                                                           ; 2. Haelfte mit!
          (move-view-to-object              part (* time 0.1))
          (zoom-view-to-motion-and-object   part motion (* time 0.1))
          (mixed-show-object-move           part motion (* time 0.7)))

(anim-rule show-explosion-irt     (part whole time) ; weg und zum Objekt fahren
          :sequential             ; und  und Explosion mit fester Kamera zeigen
          (zoom-aim-to-object     whole (* time 0.2))
          (go-aim-to-object       part (* time 0.2))
          (show-object-explode    part (* time 0.6)))
```



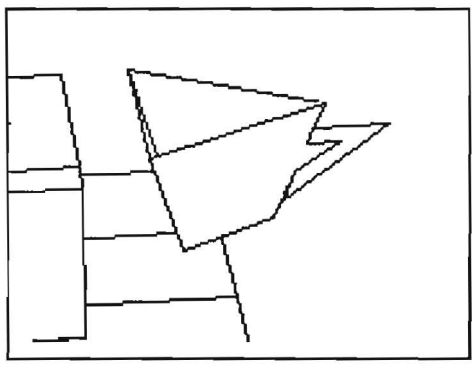


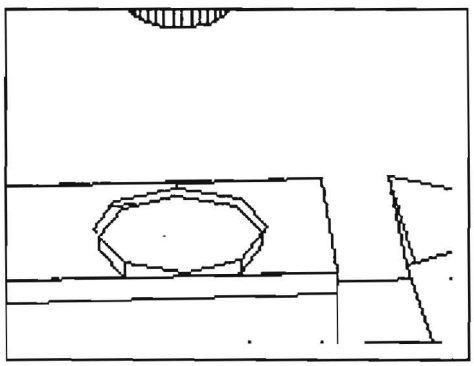


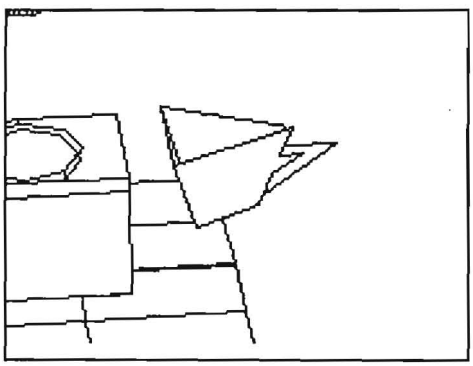


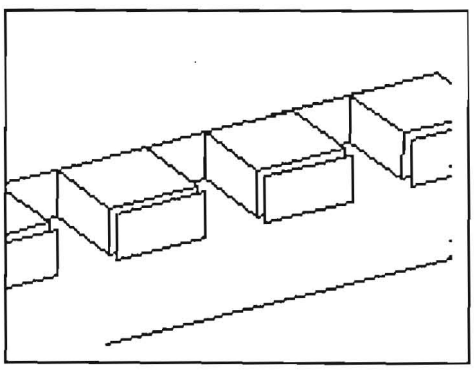


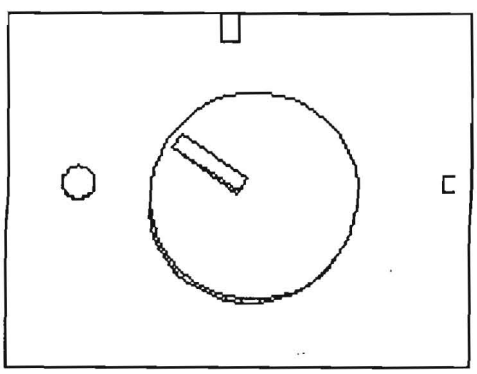
```
;;; Lokalisation und Bewegungen ohne Umschweife (dir = directly)

(anim-rule localize-object-dir    (part time)        ; Objekt direkt lokalisieren
          :sequential
          (zoom-aim-to-object     part (* time 0.7))
          (hold-view              (* time 0.3)))

(anim-rule show-motion-dir        (part motion time) ; direkt zum Objekt und
          :sequential                                ; Bewegung mit fester Kamera
          (zoom-view-to-motion-and-object part motion (* time 0.3)) ; zeigen
          (show-object-move               part motion 0 1 (* time 0.7)))

(anim-rule show-motion-following-dir (part motion time) ; direkt zum Objekt und
          :sequential             ; Bewegung zeigen, Zielpunkt mit
          (zoom-view-to-object-and-motion part motion (* time 0.3))
          (follow-object-move             part motion 0 1 (* time 0.7)))

(anim-rule show-explosion-dir     (part time)         ; direkt zum Objekt und
          :sequential                                 ; Expl. mit fester Kamera
          (go-aim-to-object       part (* time 0.2))
          (show-object-explode    part (* time 0.8)))

;;; Lokalisation, Bewegungen und Explosion in Relation zum Oberobjekt:
;;; (keine Extension)

(anim-rule localize-object-sup    (part time)        ; Objekt in Rel. zum O.o.
          :sequential                                 ; lokalisieren
          (zoom-aim-to-object     (superobject part) (* time 0.2))
          (move-view-to-object    part (* time 0.3))
          (zoom-to-object         part (* time 0.3))
          (hold-view              (* time 0.2)))

(anim-rule show-motion-sup        (part motion time) ; weg und zum Objekt und
          :sequential                                 ; Bewegung mit fester Kamera
          (zoom-aim-to-object     (superobject part) (* time 0.1))
          (move-view-to-object    (superobject part) (* time 0.1))
          (zoom-view-to-motion-and-object part motion (* time 0.1))
          (show-object-move       part motion 0.0 1.0 (* time 0.7)))
```



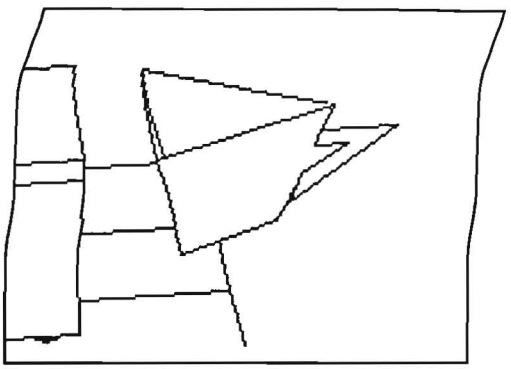


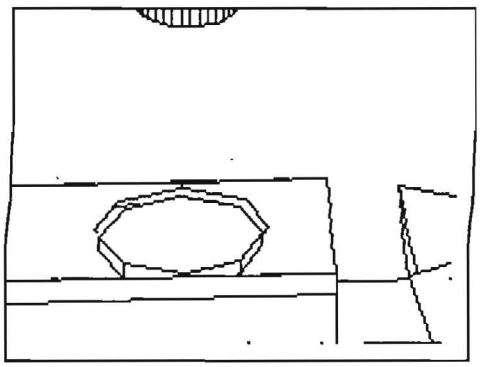


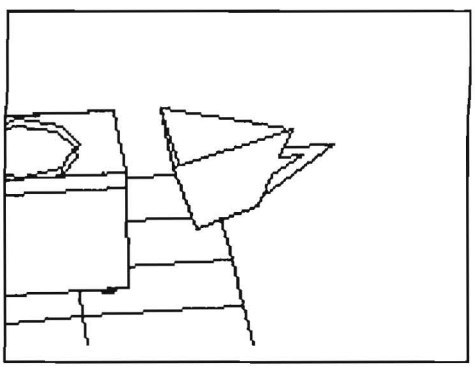


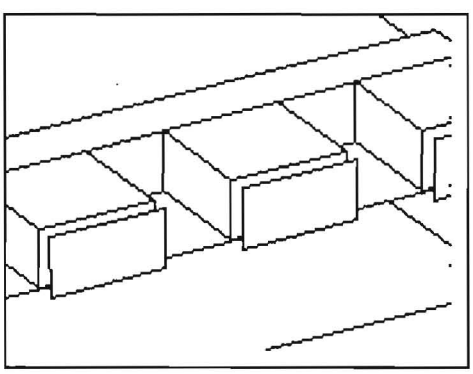


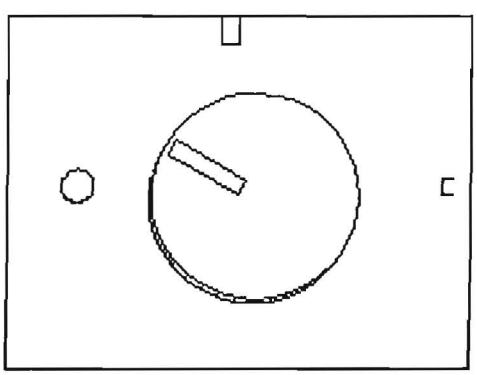
```
(anim-rule show-motion-following-sup (part motion time) ; weg und zum Objekt und
          :sequential                                    ; Bewegung, Zielpunkt mit
          (zoom-aim-to-object    (superobject part) (* time 0.1))
          (move-view-to-object   part (* time 0.1))
          (zoom-view-to-object-and-motion part motion (* time 0.1))
          (follow-object-move    part motion 0.0 1.0 (* time 0.7)))

(anim-rule show-motion-mixed-sup  (part motion time)  ; weg und zum Objekt und
          :sequential                                 ; Bewegung, Zielpunkt
          (zoom-aim-to-object    (superobject part) (* time 0.1)) ; in der 2.
          (move-view-to-object   part (* time 0.1))               ; Haelfte mit
          (zoom-view-to-motion-and-object part motion (* time 0.1))
          (mixed-show-object-move         part motion (* time 0.7)))

(anim-rule show-explosion-sup     (part time)     ; weg und zum Objekt und
          :sequential                             ; Explosion mit fester Kamera
          (zoom-aim-to-object    (superobject part) (* time 0.2))
          (go-aim-to-object      part (* time 0.2))
          (show-object-explode   part (* time 0.6)))

;;; Kontextgesteuerte Regeln:

(anim-rule localize-cond-1 (part time)
;;; aufs Superobjekt fahren und wieder zurueck + halten
          :sequential
          (zoom-aim-to-object      (superobject part) (* time 0.4))
          (hold-view               (* time 0.1))
          (zoom-aim-to-object part (* time 0.4))
          (hold-view               (* time 0.1)))

(anim-rule localize-cond-2 (part time)
;;; vom Superobjekt aufs Objekt fahren und halten
          :sequential
          (zoom-view-to-object part (* time 0.8))
          (hold-view                (* time 0.2)))

(anim-rule localize-cond-3 (part time)
;;; vom Superobjekt aufs Objekt zoomen und halten
          :sequential
          (zoom-aim-to-object part (* time 0.8))
          (hold-view          (* time 0.2)))
```



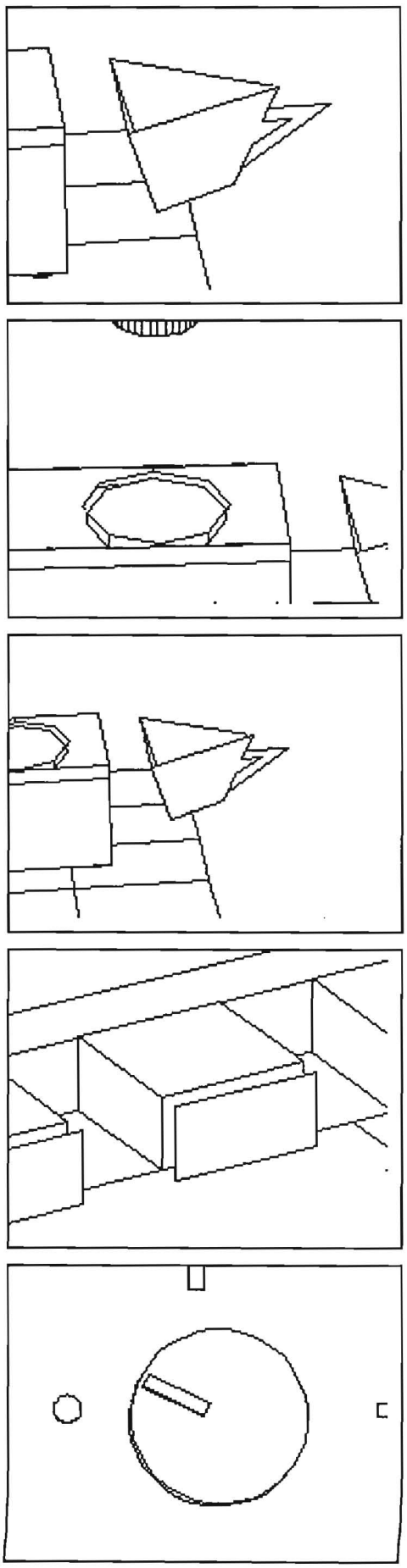




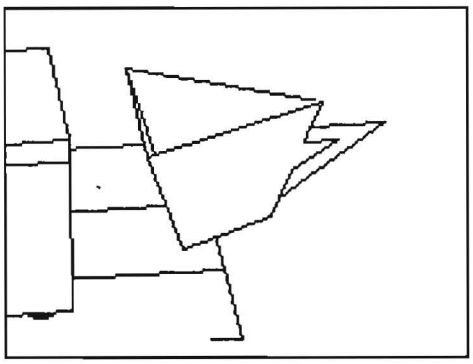


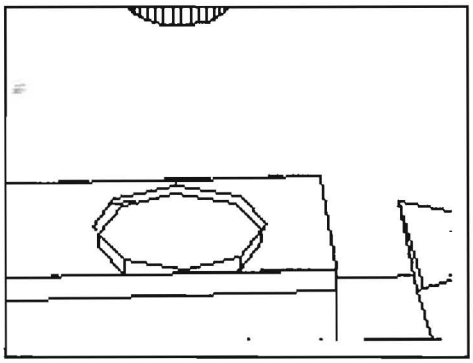


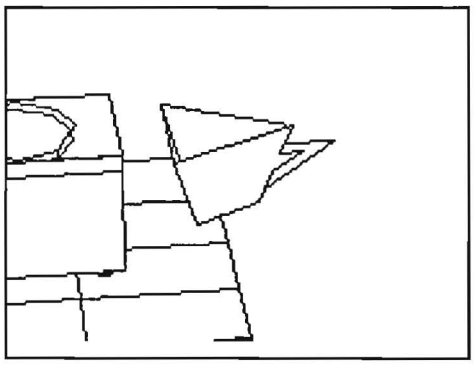
```
(anim-rule localize-cond-4 (part time)
;;; vom Superobjekt aufs Objekt fahren und halten
          :sequential
          (move-view-to-object      part (* time 0.0))
          (zoom-to-object           (superobject part) (* time 0.0))
          (hold-view                (* time 0.1))
          (zoom-to-object           part  (* time 0.7))
          (hold-view                (* time 0.2)))

(anim-rule show-motion-cond-1 (part motion time)
;;; Schnitt auf Objekt und Bewegung und los!
          :sequential
          (cut-view-to-motion-and-object part motion (* time 0.1))
          (mixed-show-object-move        part motion (* time 0.9)))

(anim-rule show-motion-cond-2 (part motion time)
;;; Schwenk auf Objekt und Bewegung und los!
          :sequential
          (zoom-view-to-motion-and-object part motion (* time 0.2))
          (mixed-show-object-move         part motion (* time 0.8)))

(anim-rule show-motion-cond-3 (part motion time)
;;; Schnitt aufs Superobjekt, dann Schwenk aufs Objekt und los!
          :sequential
          (move-view-to-object            part (* time 0.0))
          (zoom-to-object                 (superobject part) (* time 0.0))
          (hold-view                      (* time 0.1))
          (zoom-view-to-motion-and-object part motion (* time 0.2))
          (mixed-show-object-move         part motion (* time 0.7)))

(anim-rule show-explosion-cond-1 (part time)
;;; zoom auf Superobjekt und los!
          :sequential
          (zoom-view-to-object (superobject part) (* time 0.3))
          (show-object-explode part (* time 0.7)))

(anim-rule show-explosion-cond-2 (part time)
;;; Schwenk auf Objekt und Bewegung und los!
          :sequential
          (show-object-explode part (* time 1)))
```

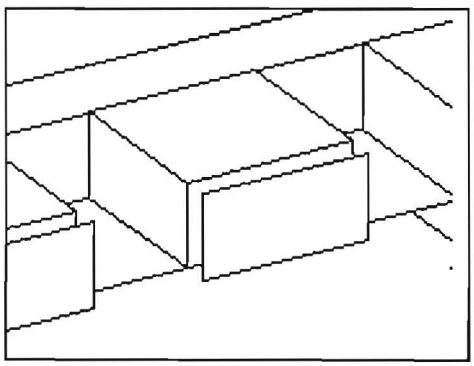


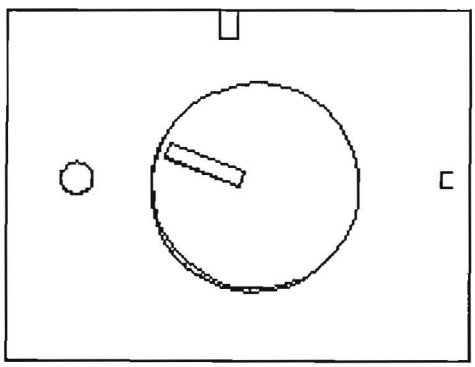
```
(anim-rule show-explosion-cond-3 (part time)
;;; Schnitt aufs Superobjekt, dann Schwenk aufs Objekt und los!
          :sequential
          (cut-view-to-object       (superobject part) (* time 0.1))
          (show-object-explode part (* time 0.9)))

(anim-rule localize-object (part time)
          :cond
          ((and (camera cam-pos part)
                (camera aim-pos part)
                (camera va-for  part))
;;; Kameraeinstellung: alles schon auf dem Objekt.
           (localize-cond-1 part time))
          ((and (camera cam-pos (superobject part))
                (camera aim-pos (superobject part))
                (camera va-for  (superobject part)))
;;; alles auf dem Superobjekt
           (localize-cond-2 part time))
          ((camera cam-pos (superobject (superobject part)))
           (localize-cond-3 part time))
          (t                                          ; alles andere
            (localize-cond-4 part time)))

(anim-rule show-object-motion (part motion time)
          :cond
          ((and (camera cam-pos part)
                (camera aim-pos part))
;;; Kameraeinstellung: alles schon auf dem Objekt.
           (show-motion-cond-1 part motion time))
          ((and (camera cam-pos (superobject part))
                (camera aim-pos (superobject part))
                (camera va-for  (superobject part)))
;;; alles auf dem Superobjekt
           (show-motion-cond-2 part motion time))
          (t                                          ; alles sonst
           (show-motion-cond-3 part motion time)))
```



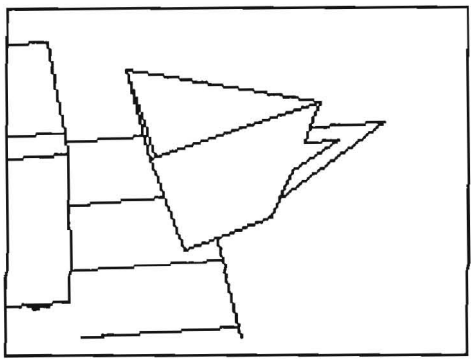


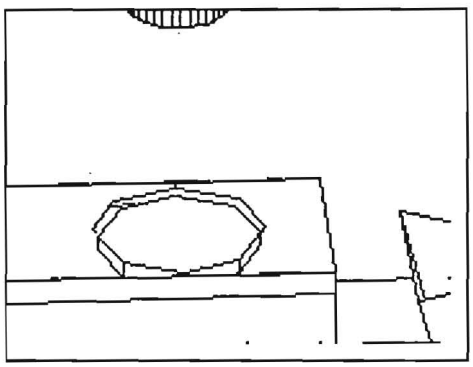


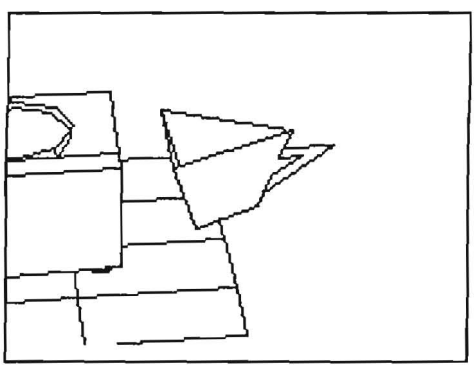


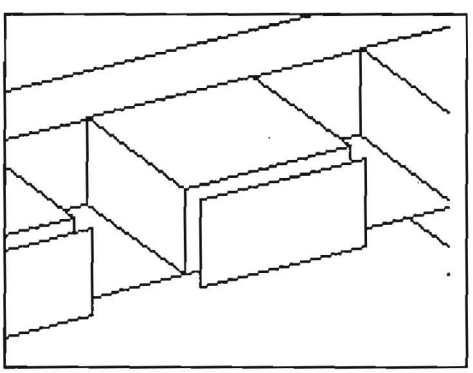


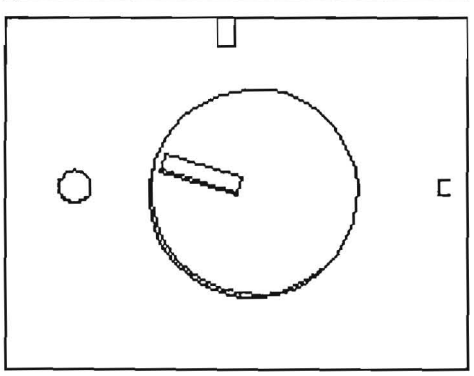
```
(anim-rule show-object-explosion (part time)
           :cond
           ((and (camera cam-pos part)
                 (camera aim-pos part)
                 (camera va-for  part))
;;; Kameraeinstellung: alles schon auf dem Objekt.
            (show-explosion-cond-1 part time))
           ((and (camera cam-pos (superobject part))
                 (camera aim-pos (superobject part))
                 (camera va-for  (superobject part)))
;;; alles auf dem Superobjekt
            (show-explosion-cond-2 part time))
           (t                                          ; alles sonst
            (show-explosion-cond-3 part time)))
```







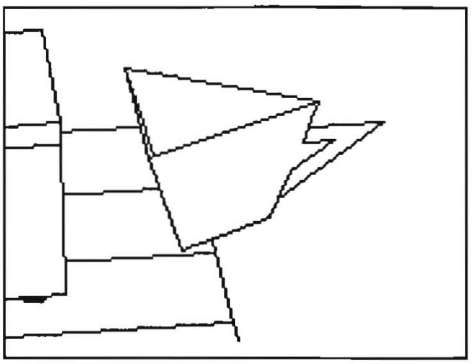


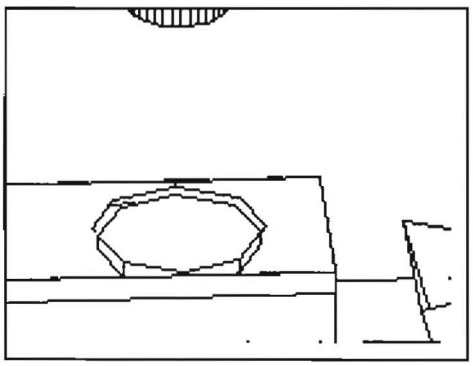


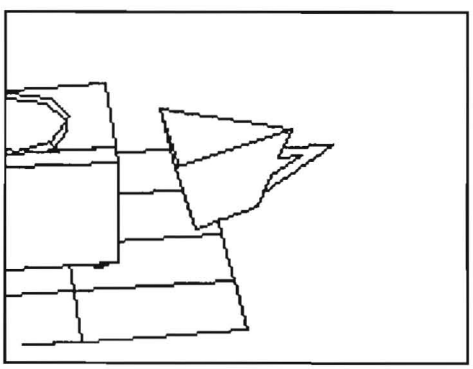
B Details des Animationsplaners

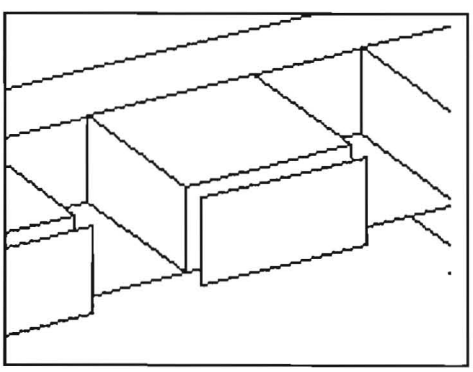
Das Herzstück des hierarchischen Animationsplaners: eine mehrfache Rekursion mit rekursiven Hilfsfunktionen, die aus einem Visualisierungsziel auf oberster Abstraktionsebene ein Script in der vorne definierten Zwischensprache generiert:

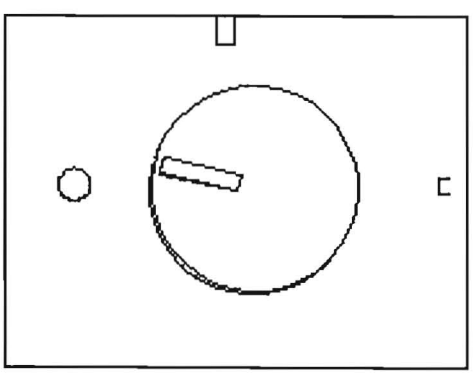
```
;;; Dekomposition eines nicht-primitiven Ziels, dreifache Rekursion
;;; mit rekursiven Hilfsfunktionen...

;;; Rekursionsanfang:
(defmethod (decompose-goal animation-planer)
           (goal)
  (let* ((rule (find-anim-rule (first goal)))
         (varnames (second rule))                          ; formale Variablen
         (varvalues (cdr goal))                            ; zugehoerige Werte
         (varassoc  (mapcar #'list varnames varvalues))); Zuordnung Name-->Wert

    (cond                                      ; Verschiedene Typen von Regeln:

;;; Ziel enthaelt eine kontextabhaengige Verzweigung:
;;; Verzweigung berechnen und gleich rekursiv weiter
      ((eq (third rule) ':cond)
       (loop for pair in (fourth rule)
             when (eval-condition
                    self (first pair) varassoc)
               return (decompose-goal
                        self
                        (cons (car (second pair))
; Schleife ueber alle Argumente
                              (loop for arg in (cdr (second pair))
                                    collect
                                      (eval-argument self arg varassoc)
                                      )))))

;;; Ziel ist noch kein Primitiv und muss weiter dekomponiert werden:
      ((not (assoc (car goal) primitives))
       (instantiate-goal self goal))

;;; Ziel ist ein Primitiv, kann so uebernommen werden
;;; Aktualisierung des Kontextes:
```



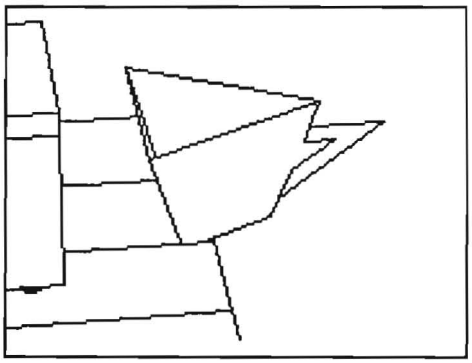


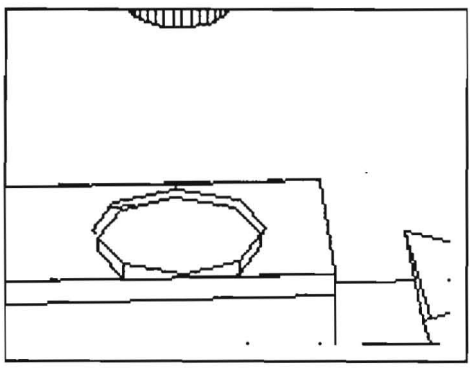


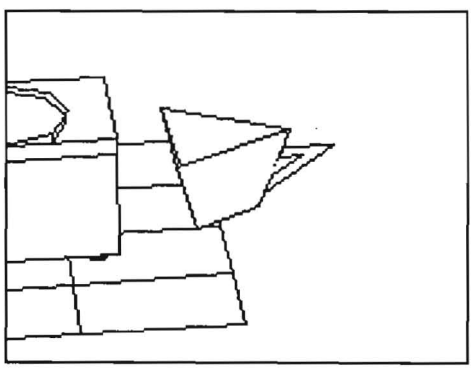


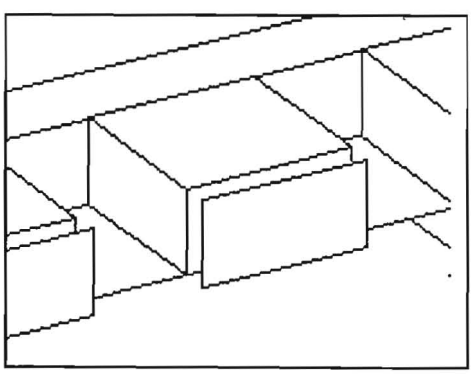


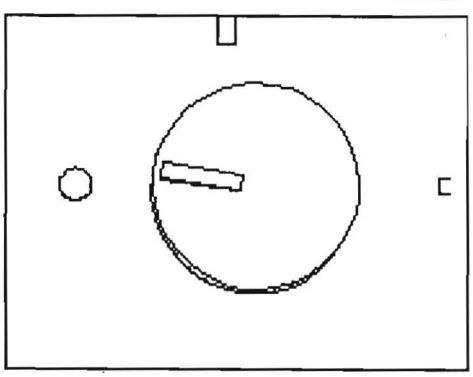
```
      (t
       (let* ((varnames
                (second (assoc (car goal)
                               primitives))) ; formale Variablen
              (varvalues (cdr goal)) ; zugehoerige Werte
              (varassoc  (mapcar #'list varnames varvalues)); Zuordnung Name-->Wert
              (part (betty-object-name (second (assoc 'part varassoc)))))
;; die versch. Primitive:
         (select (car goal)
           (('move-cam-to-object)
;            (when (check-object-state self 'camera 'cam-pos part)
;              (format *betty-trace-stream*
;                      "~&BETTY-Warning: obsolete camera-motion"))
            (change-object-state self 'camera 'cam-pos part)
            (format *betty-trace-stream*
                    "~&BETTY: cam in pos. for ~a" (betty-object-name part)))

           (('move-aim-to-object)
;            (when (check-object-state self 'camera 'aim-pos part)
;              (format *betty-trace-stream*
;                      "~&BETTY-Warning: obsolete aimpoint-motion"))
            (change-object-state self 'camera 'aim-pos part)
            (format *betty-trace-stream*
                    "~&BETTY: aimpoint at ~a" (betty-object-name part)))

           (('adjust-va-for-object)
;            (when (check-object-state self 'camera 'va-for part)
;              (format *betty-trace-stream*
;                      "~&BETTY-Warning: obsolete zoom"))
            (change-object-state self 'camera 'va-for part)
            (format *betty-trace-stream*
                    "~&BETTY: view-angle for ~a"
                    (betty-object-name part)))

           (('move-object)
            (when (check-object-state self 'camera 'aim-pos part)
              (change-object-state self 'camera 'aim-pos nil)
              (format *betty-trace-stream*
                      "~&BETTY: aimpoint unspecific")))

           (('explode-object)
```



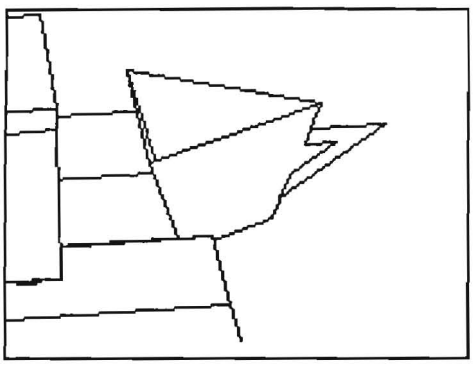


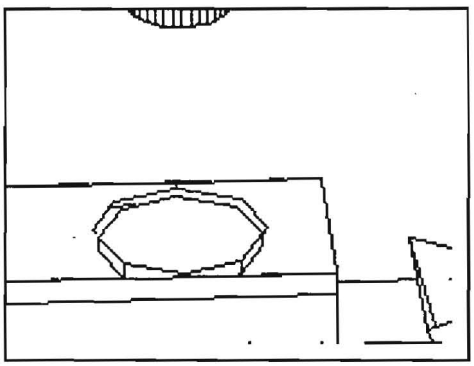


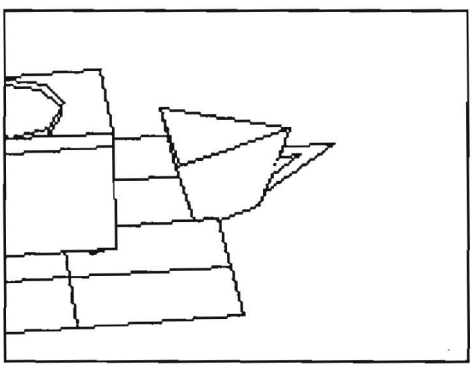


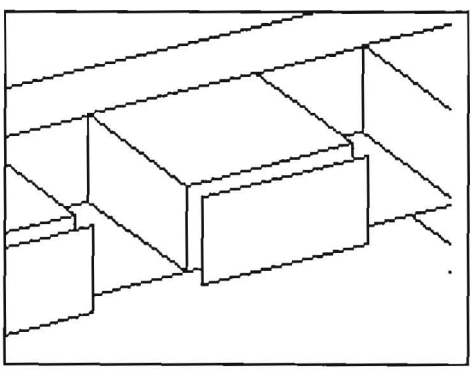


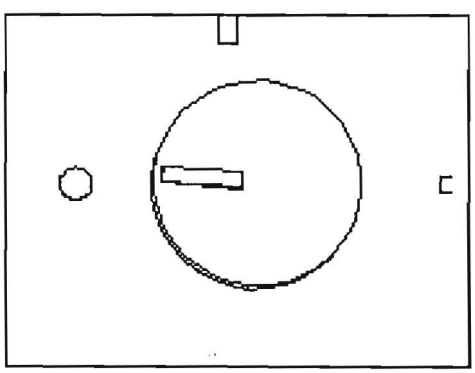
```
            (when (check-object-state self 'camera 'va-for part))
            (change-object-state self 'camera 'va-for nil)
            (format *betty-trace-stream*
                    "~&BETTY: view-angle unspecific"))))
;;; und Returnwert:
       goal))))


;;; Hilfsfunktion zum auswerten der Bedingungen:
(defmethod (eval-condition animation-planer)
           (condition varassoc)
  (cond ((eq condition t)
         t)
        ((member (first condition)
                 '(and or not))
         (eval `(,(first condition)
                 ,@(loop for subcondition in (cdr condition); Rek. f. geschachtelte
                         collect (eval-condition self
                                                 subcondition varassoc))))); Bedingungen
        (t   (format *betty-trace-stream* "~&BETTY: testing (~a ~a ~a)"
                     (first condition) (second condition)
                     (betty-object-name (eval-argument self
                                                       (third condition) varassoc)))
             (check-object-state self
                                 (first condition)
                                 (second condition)
                                 (eval-argument self (third condition) varassoc)
                                 ))))

;;; Hilfsfunktion zur Auswertung der Argumente:
(defmethod (eval-argument animation-planer)
           (arg varassoc)
  (cond ((null arg) arg)
        ((numberp arg) arg)
        ((atom arg)
         (cadr (assoc arg varassoc)))
; Argument ist ein Atom
        ((member (first arg) '(* + / - max min))
         (funcall (first arg)                  ; Argument ist ein Term
                  (if (numberp (second arg)) ; Zahl oder Variable ?
                      (second arg)
```



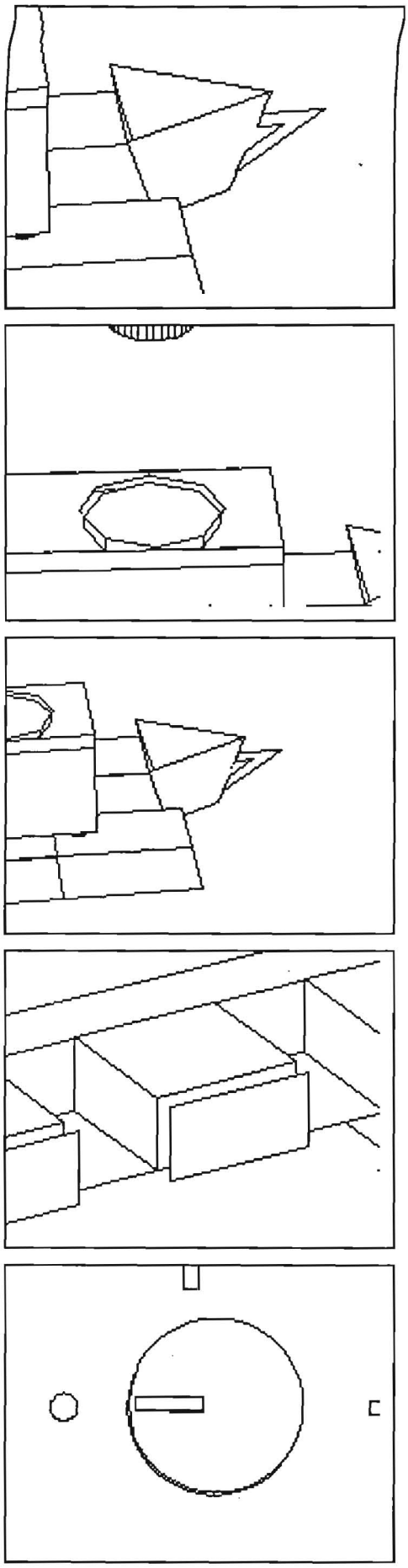




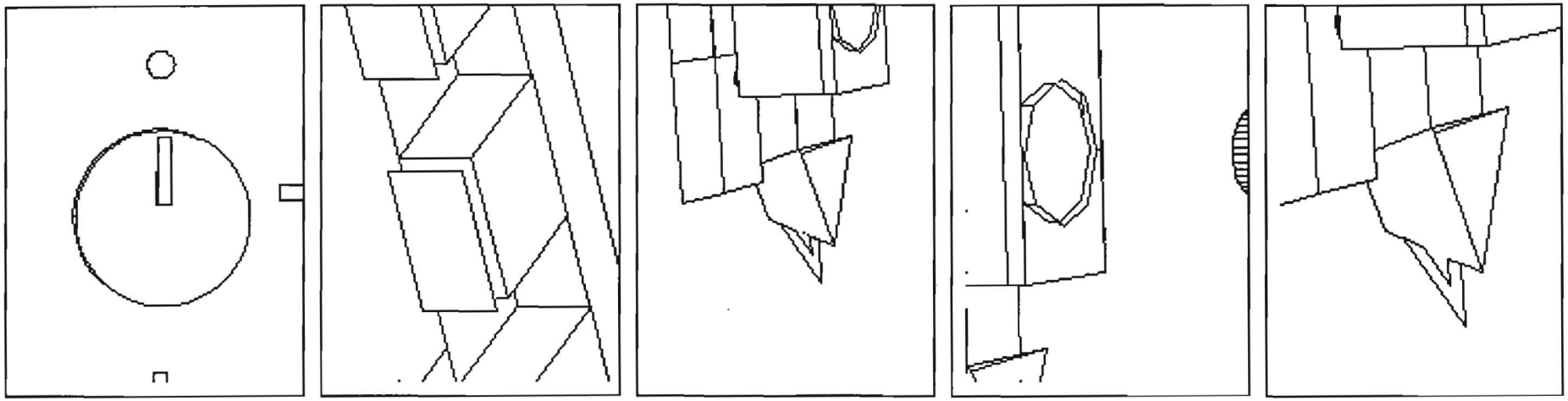




```
                    (cadr (assoc (second arg) varassoc)))
                (if (numberp (third arg)) ; Zahl oder Variable ?
                    (third arg)
                    (cadr (assoc (third arg) varassoc)))))
       ((equal (first arg) 'superobject)
; reservierte Funktion: superobject
        (betty-find-super-object (eval-argument self (second arg) varassoc)))
        (t                                  ; Argument ist eine Liste
         (loop for element in arg
               collect (eval-argument self element varassoc)))))

;;; Rekursionsschritt:
(defmethod (instantiate-goal animation-planer)
           (goal)
  (let*
    ((rule (or (find-anim-rule (car goal))
               (format *betty-trace-stream*
                       "~&There is no rule for ~a~%" (car goal))))
     (varnames  (second rule))                ; formale Variablen
     (varvalues (cdr goal))                 ; zugehoerige Werte
     (ordering  (third rule))                 ; :sequential oder :parallel
     (subgoals  (fourth rule))                ; Liste der Unterziele
     (varassoc  (mapcar #'list varnames varvalues)); Zuordnung Name-->Wert
     (instantiated-subgoals                         ; subgoals m. Werten st. Namen
       (loop for subgoal in subgoals                ; Schleife ueber alle subgoals
             collect
               (cons (car subgoal)
                     (loop for arg in (cdr subgoal); Schleife ueber alle Argumente
                           collect
                             (eval-argument self arg varassoc)
                             )
                     )
               )
       )
     (dec-inst-subgoals           ; fertig inst. subgoals weiter dekomponieren
       (loop for goal in instantiated-subgoals
             collect
               (decompose-goal self goal)))) ; Rekursion...
    (and goal (list ordering dec-inst-subgoals))))

;;; Umsetzung eines dekomponierten Ziels in ein Scriptstueck
```



```
(defmethod (convert-decomposed-goal-to-script animation-planer)
           (goal &optional (now-time actual-time))
  (or (and (assoc (car goal) primitives)
           (translate-primitiv self goal))
      (let* ((ordering (car goal))
             (subgoals (cadr goal))
             (starttime now-time)
             (endtime starttime)
             (script nil)
             (script-endtime now-time)
             (translation (list ordering)))
        (if (eq ordering ':sequential)
;; Sequenziell:
            (loop for subgoal in subgoals do
              (setq script
                    (convert-decomposed-goal-to-script self subgoal endtime))
              (setq script-endtime
                    (second (assoc ':end-time script)))
              (if (> script-endtime endtime)
                  (setq endtime script-endtime))
              (setq translation
                    (append translation (list script))))
;; Parallel:
            (loop for subgoal in subgoals do
              (setq script (convert-decomposed-goal-to-script
                            self subgoal starttime))
              (setq script-endtime
                    (second (assoc ':end-time script)))
              (if (> script-endtime endtime)
                  (setq endtime script-endtime))
              (setq translation
                    (append translation (list script)))
                 finally (setq actual-time endtime)))
        (if (> endtime overall-end-time)
            (setq overall-end-time endtime))
        (setq translation
              (append `((:start-time ,starttime)
                        (:end-time ,endtime))
                      (list translation))))))
```



Literatur

[Al85] J.F. Allen, P.J. Hayes: „A common-sense theory of time"in: „Proceedings of the 9th International Joint Conference on Artificial Intelligence", 528-531, Los Angeles Ca, 1985

[An90] Elisabeth André, Thomas Rist: „Wissensbasierte Perspektivenwahl für die automatische Erzeugung von 3D-Objektdarstellungen"in K. Kansy, P.Wißkirchen (Ed.): „Graphik und KI", 48-57, Springer Verlag Berlin, Heidelberg 1990

[An92] Elisabeth André, Wolfgang Finkler, Winfried Graf, Thomas Rist, Anne Schauder: „WIP: The Automatic Synthesis of Multimodal Presentations", DFKI Saarbrücken 1992 (RR-92-46)

[An93] Elisabeth André, Thomas Rist: „The Design of illustrated Documents as a Planning Task"in: M. Maybury (Ed.): „Intelligent Multimedia Interfaces"94-116, AAAI Press 1993, auch als DFKI (RR-92-45)

[AVI92] M. F. Costabile, T. Catarci, S. Levialdi: „Advanced Visual Interfaces, Proceedings of the International Workshop AVI'92", World Scientific Singapore 1992, ISBN 981-02-1123-6

[Ba90] Norman I.Badler, Bonnie L. Webber, Jugal Kalita, Jeffrey Esakov: „Animation from Instructions"in [BBZ90], 50-93

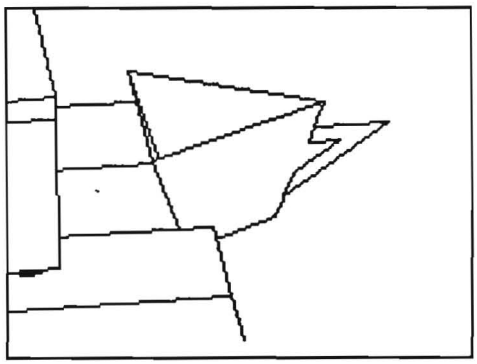
[BBZ90] Norman Badler, Brian Barsky, David Zeltzer: „Making Them Move: Mechanics, Control and Animation of Articulated Figures", Morgan Kaufmann 1990

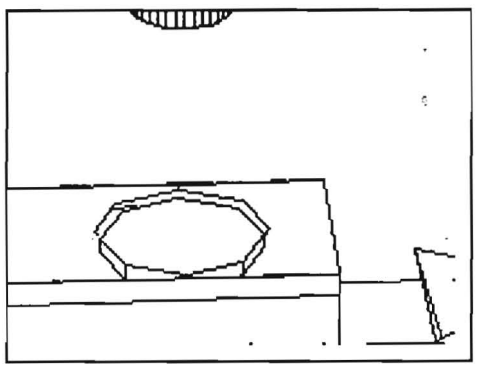
[BPW93] Norman Badler, Cary Phillips, Bonnie Webber: „Simulating Humans: Computer Graphics Animation and Control", Oxford University Press, New York / Oxford 1993, ISBN 0-19-507359-2

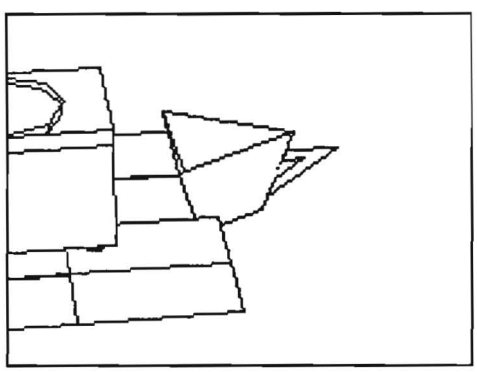
[Br94] Michael Braun: „Ein Berechungsmodell zur prozeduralen Beschreibung von Computeranimationen", Diplomarbeit Universität des Saarlandes 1994

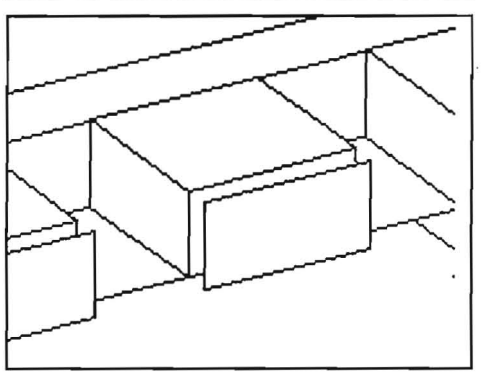
[Du93] Dunker, Achim: „Die chinesische Sonne scheint immer von unten, Licht- und Schattengestaltung im Film", TR-Verlagsunion, München 1993, ISBN 3-8058-2700-8

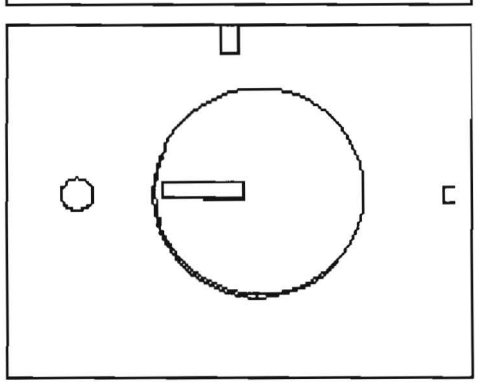
[Fe89] Steven K. Feiner: „Specifying Composite Illustrations with Communicative Goals", Proc. UIST '89 (ACM SIGGRAPH Symposium on User Interface Software and Technology) Williamsburg VA, Nov 13-15 1989, 1-9











[Fi71] R. E. Fikes and N.J. Nilsson: „STRIPS: A new approach to the application of theorem proving to problem solving", Artificial Intelligence, 2:189-208, 1971

[Fo90] James D. Foley, Andries van Dam, John F. Hughes, Steven K. Feiner: „Computer Graphics: Principles and Practice", 2nd Edition 1990: Addison-Wesley

[Gr92] Winfried Graf, Markus A. Thies: „Perspektiven zur Kombination von automatischem Animationsdesign und planbasierter Hilfe", DFKI Saarbrücken 1992

[Gr93] Winfried H. Graf: „LAYLAB: A Constraint-Based Layout Manager for Mulitmedia Presentations", DFKI Saarbrücken 1993, (RR-93-41)

[Ha91] Karin Harbusch, Wolfgang Finkler, Anne Schauder: „Incremental Syntax Generation with Tree Adjoining Grammars", DFKI Saarbrücken 1991, (RR-91-25)

[Ka26] Wassilly Kandinsky: „Punkt und Linie zu Fläche", Bauhaus-Bücher Bd. 9, Verlag Albert Langen, München 1926

[Ka90] Peter Karp, Steven Feiner: „Issues in the Automated Generation of Animated Presentations", Proc. Graphics Interface 1990, 39-48

[Ka93] Peter Karp, Steven Feiner: „Automated Presentation Planning of Animation using Task Decomposition with Heuristic Reasonig", Proc. Graphics Interface 1993, 118-127

[Kr91] A.Krüger, G.Schneider, F.Schneiderlöchner, C.Schommer, „Wissensbasierte Graphikgenerierung", Abschlußbericht FOPRA WS 90/91, Universität des Saarlandes, Saarbrücken 1991

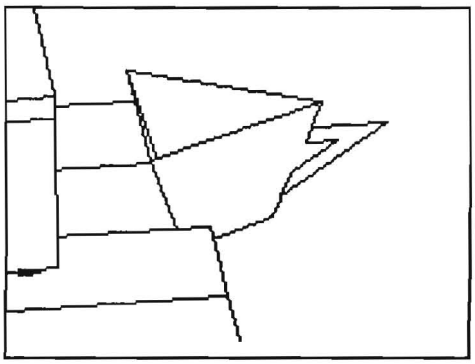
[Kr94] Antonio Krüger: „PROXIMA: Ein System zur wissensbasierten Generierung von Abstraktionen in technischen Gebrauchsgrafiken", DFKI Saarbrücken 1994

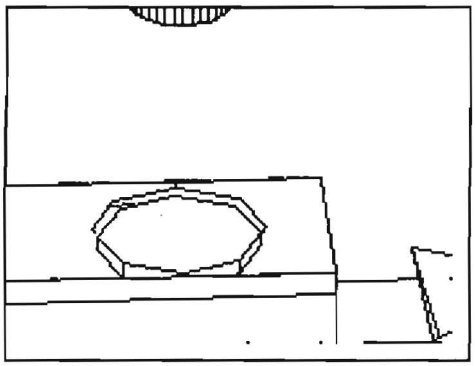
[Ma75] Jörg Michael Matthaei: „Grundfragen des Grafik-Design", Heinz Moos Verlag München, 1975, ISBN 3-7879-0081-0

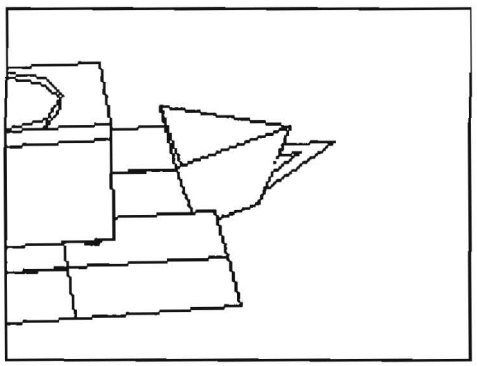
[Ma88] Jost J. Marchesi: „Fotokollegium", Bd. 1, Verlag PHOTOGRAPHIE, CH-8201 Schaffhausen 1988, ISBN 3-7231-6000-X

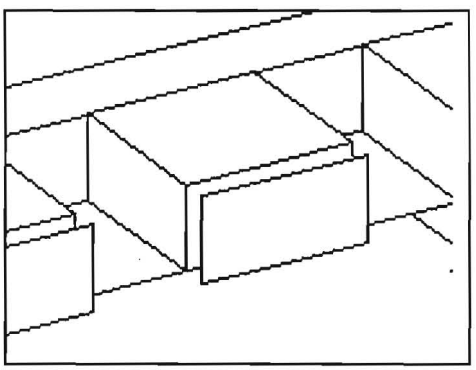
[Mc69] J. McCarthy, P.J. Hayes: „Some Philosophical Problems from the Standpoint of Artificial Intelligence"in B. Meltzer, D. Mitchie (Ed.): „Machine Intelligence 4", 463-502, Edinburgh University Press, Edinburgh, UK, 1969

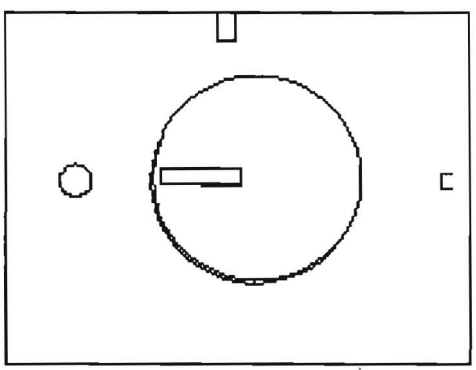
[Pe91] Catherine Pelachaud: „Communication and coarticulation in facial animation", PhD Thesis, Computer and Information Science, Univ. of Pennsylvania, Philadelphia 1991, Tech. Report MS-CIS-91-82











[Pe92] Catherine Pelachaud: „Functional Decomposition of Facial Expressions for an Animation System", in [AVI92], 26-49

[PPP] Elisabeth André, Winfried Graf, Jochen Heinsohn, Bernhard Nebel, Hans-Jürgen Profitlich, Wolfgang Wahlster: „PPP - Personalized Plan-Based Presenter", DFKI Saarbrücken 1993

[Ri92] Thomas Rist, Elisabeth André: „Incorporating Graphics Design and Realization into the Multimodal Presentation System WIP ", in [AVI92], 193-207

[Sc92] Anne Schauder: „Incremental Syntactic Generation of Natural Language with Tree Adjoining Grammars ", DFKI Saarbrücken 1992, (D-92-21)

[Sc93] Georg Schneider, Antonio Krüger: „Eine Werkbank zur Erzeugung von 3D-Illustrationen", Proceedings zum GI-Workshop „Computergrafik und automatisierte Layoutsynthese", Gesellschaft für Informatik 1993

[Sc94] Georg Schneider: „TOPAS: Eine Werkbank zur Erzeugung von 3D-Illustrationen", DFKI Saarbrücken 1994

[Si94] Jörg Schirra: „Bildbeschreibung als Verbindung von visuellem und sprachlichem Raum", Dissertation, Universität des Saarlandes 1994

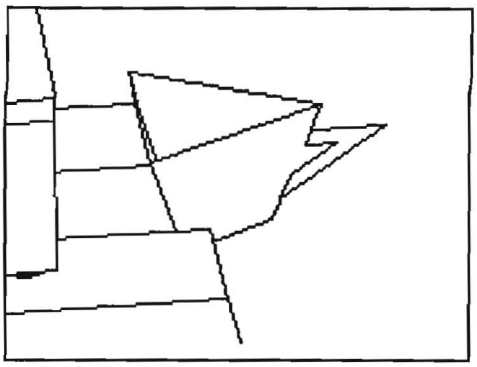
[St93] Eva Stopp: „GEO-ANTLIMA: Konstruktion dreidimensionaler mentaler Bilder aus sprachlichen Szenenbeschreibungen", Diplomarbeit, Universität des Saarlandes SFB 314, Saarbrücken 1993, ISSN 0944-7822

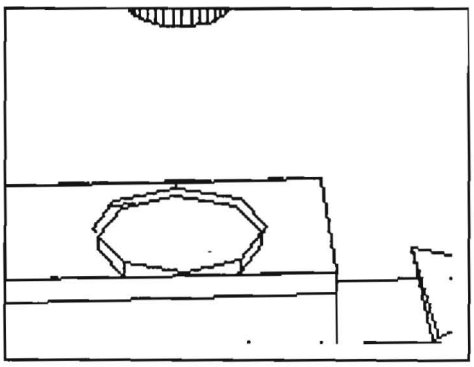
[Wa91] W. Wahlster, E. André, S. Bandyoppadhyay, W.Graf, Th. Rist: „WIP: The Coordinated Generation of Multimodal Presentations from a Common Representation"in A.Ortony, J.Slack, O. Stock (Ed.): Communication from an Artificial Intelligence Perspective: Theoretical and applied Issues, Springer Heidelberg 1992, 121 - 144, auch als DFKI (RR-91-08)

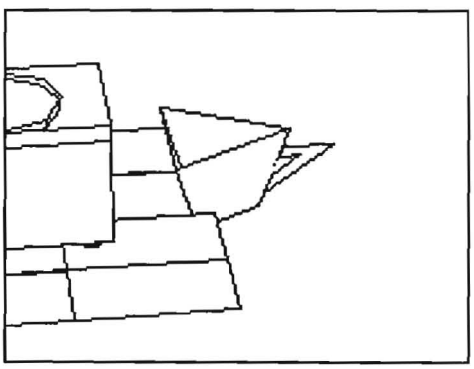
[Wa93] Wolfgang Wahlster, Elisabeth André, Wolfgang Finkler, Hans-Jürgen Profitlich, Thomas Rist: „WIP : Plan-based Integration of Natural Language and Graphics Generation",Artificial Intelligence 63 (1993) 387-427

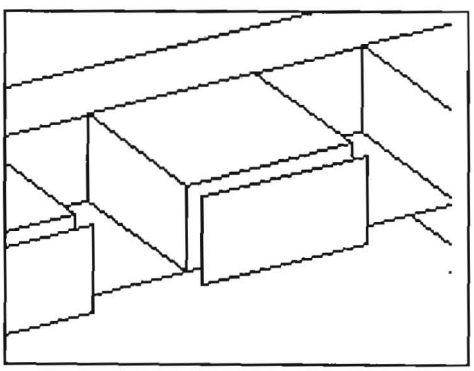
[We92] Bonnie Webber, Norman Badler et al.: „Doing What You're told: Following Task Instructions in Changing, but Hospitable Environments"University of Pennsylvania Philadelphia, PA 19104-6389, September 1992

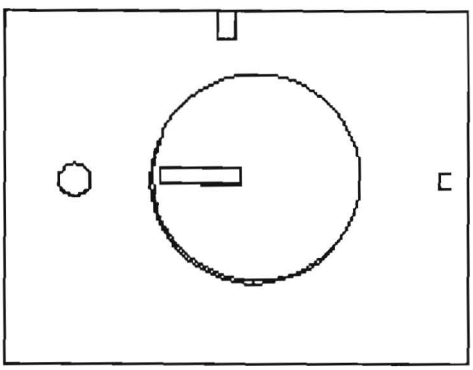
[Ze90] David Zeltzer: „Task-level Graphical Simulation: Abstraction, Representation, and Control"in [BBZ90], 3-33











Deutsches
Forschungszentrum
für Künstliche
Intelligenz GmbH

-Bibliothek, Information
und Dokumentation (BID)-
PF 2080
67608 Kaiserslautern
FRG

Telefon (0631) 205-3506
Telefax (0631) 205-3210

e-mail
dfkibib@dfki.uni-kl.de
WWW
http://www.dfki.uni-
sb.de/dfkibib

Veröffentlichungen des DFKI

Die folgenden DFKI Veröffentlichungen sowie die aktuelle Liste von allen bisher erschienenen Publikationen können von der oben angegebenen Adresse oder per anonymous ftp von ftp.dfki.uni-kl.de (131.246.241.100) im Verzeichnis pub/Publications bezogen werden.
Die Berichte werden, wenn nicht anders gekennzeichnet, kostenlos abgegeben.

DFKI Publications

The following DFKI publications or the list of all published papers so far are obtainable from the above address or by anonymous ftp from ftp.dfki.uni-kl.de (131.246.241.100) in the directory pub/Publications.
The reports are distributed free of charge except where otherwise noted.

DFKI Research Reports

RR-94-39
Hans-Ulrich Krieger
Typed Feature Formalisms as a Common Basis for Linguistic Specification.
21 pages

RR-94-38
Hans Uszkoreit, Rolf Backofen, Stephan Busemann, Abdel Kader Diagne, Elizabeth A. Hinkelman, Walter Kasper, Bernd Kiefer, Hans-Ulrich Krieger, Klaus Netter, Günter Neumann, Stephan Oepen, Stephen P. Spackman.
DISCO–An HPSG-based NLP System and its Application for Appointment Scheduling.
13 pages

RR-94-37
Hans-Ulrich Krieger, Ulrich Schaefer.
TDL - A Type Description Language for HPSG, Part 1: Overview.
54 pages

RR-94-36
Manfred Meyer
Issues in Concurrent Knowledge Engineering. Knowledge Base and Knowledge Share Evolution.
17 pages

RR-94-35
Rolf Backofen
A Complete Axiomatization of a Theory with Feature and Arity Constraints
49 pages

RR-94-34
Stephan Busemann, Stephan Oepen, Elizabeth A. Hinkelman, Günter Neumann, Hans Uszkoreit
COSMA - Multi-Participant NL Interaction for Appointment Scheduling
80 pages

RR-94-33
Franz Baader, Armin Laux
Terminological Logics with Modal Operators
29 pages

RR-94-31
Otto Kühn, Volker Becker, Georg Lohse, Philipp Neumann
Integrated Knowledge Utilization and Evolution for the Conservation of Corporate Know-How
17 pages

RR-94-23
Gert Smolka
The Definition of Kernel Oz
53 pages

RR-94-20
Christian Schulte, Gert Smolka, Jörg Würtz
Encapsulated Search and Constraint Programming in Oz
21 pages

RR-94-18
Rolf Backofen, Ralf Treinen
How to Win a Game with Features
18 pages

RR-94-17
Georg Struth
Philosophical Logics—A Survey and a Bibliography
58 pages

RR-94-16
Gert Smolka
A Foundation for Higher-order Concurrent Constraint Programming
26 pages

RR-94-15
Winfried H. Graf, Stefan Neurohr
Using Graphical Style and Visibility Constraints for a Meaningful Layout in Visual Programming Interfaces
20 pages

RR-94-14
Harold Boley, Ulrich Buhrmann, Christof Kremer
Towards a Sharable Knowledge Base on Recyclable Plastics
14 pages

RR-94-13
Jana Koehler
Planning from Second Principles—A Logic-based Approach
49 pages

RR-94-12
Hubert Comon, Ralf Treinen
Ordering Constraints on Trees
34 pages

RR-94-11
Knut Hinkelmann
A Consequence Finding Approach for Feature Recognition in CAPP
18 pages

RR-94-10
Knut Hinkelmann, Helge Hintze
Computing Cost Estimates for Proof Strategies
22 pages

RR-94-08
Otto Kühn, Björn Höfling
Conserving Corporate Knowledge for Crankshaft Design
17 pages

RR-94-07
Harold Boley
Finite Domains and Exclusions as First-Class Citizens
25 pages

RR-94-06
Dietmar Dengler
An Adaptive Deductive Planning System
17 pages

RR-94-05
Franz Schmalhofer, J.Stuart Aitken, Lyle E. Bourne jr.
Beyond the Knowledge Level: Descriptions of Rational Behavior for Sharing and Reuse
81 pages

RR-94-03
Gert Smolka
A Calculus for Higher-Order Concurrent Constraint Programming with Deep Guards
34 pages

RR-94-02
Elisabeth André, Thomas Rist
Von Textgeneratoren zu Intellimedia-Präsentationssystemen
22 Seiten

RR-94-01
Elisabeth André, Thomas Rist
Multimedia Presentations: The Support of Passive and Active Viewing
15 pages

RR-93-48
Franz Baader, Martin Buchheit, Bernhard Hollunder
Cardinality Restrictions on Concepts
20 pages

RR-93-46
Philipp Hanschke
A Declarative Integration of Terminological, Constraint-based, Data-driven, and Goal-directed Reasoning
81 pages

RR-93-45
Rainer Hoch
On Virtual Partitioning of Large Dictionaries for Contextual Post-Processing to Improve Character Recognition
21 pages

RR-93-44
Martin Buchheit, Manfred A. Jeusfeld, Werner Nutt, Martin Staudt
Subsumption between Queries to Object-Oriented Databases
36 pages

RR-93-43
M. Bauer, G. Paul
Logic-based Plan Recognition for Intelligent Help Systems
15 pages

RR-93-42
Hubert Comon, Ralf Treinen
The First-Order Theory of Lexicographic Path Orderings is Undecidable
9 pages

RR-93-41
Winfried H. Graf
LAYLAB: A Constraint-Based Layout Manager for Multimedia Presentations
9 pages

RR-93-40
Francesco M. Donini, Maurizio Lenzerini, Daniele Nardi, Werner Nutt, Andrea Schaerf
Queries, Rules and Definitions as Epistemic Statements in Concept Languages
23 pages

RR-93-38
Stephan Baumann
Document Recognition of Printed Scores and Transformation into MIDI
24 pages

RR-93-36
Michael M. Richter, Bernd Bachmann, Ansgar Bernardi, Christoph Klauck, Ralf Legleitner, Gabriele Schmidt
Von IDA bis IMCOD: Expertensysteme im CIM-Umfeld
13 Seiten

RR-93-35
Harold Boley, François Bry, Ulrich Geske (Eds.)
Neuere Entwicklungen der deklarativen KI-Programmierung — *Proceedings*
150 Seiten

Note: This document is available for a nominal charge of 25 DM (or 15 US-$).

RR-93-34
Wolfgang Wahlster
Verbmobil Translation of Face-To-Face Dialogs
10 pages

RR-93-33
Bernhard Nebel, Jana Koehler
Plan Reuse versus Plan Generation: A Theoretical and Empirical Analysis
33 pages

RR-93-32
David R. Traum, Elizabeth A. Hinkelman
Conversation Acts in Task-Oriented Spoken Dialogue
28 pages

RR-93-31
Elizabeth A. Hinkelman, Stephen P. Spackman
Abductive Speech Act Recognition, Corporate Agents and the COSMA System
34 pages

RR-93-30
Stephen P. Spackman, Elizabeth A. Hinkelman
Corporate Agents
14 pages

RR-93-29
Armin Laux
Representing Belief in Multi-Agent Worlds via Terminological Logics
35 pages

RR-93-28
Hans-Ulrich Krieger, John Nerbonne, Hannes Pirker
Feature-Based Allomorphy
8 pages

RR-93-27
Hans-Ulrich Krieger
Derivation Without Lexical Rules
33 pages

RR-93-26
Jörg P. Müller, Markus Pischel
The Agent Architecture InteRRaP: Concept and Application
99 pages

RR-93-25
Klaus Fischer, Norbert Kuhn
A DAI Approach to Modeling the Transportation Domain
93 pages

RR-93-24
Rainer Hoch, Andreas Dengel
Document Highlighting — Message Classification in Printed Business Letters
17 pages

RR-93-23
Andreas Dengel, Ottmar Lutzy
Comparative Study of Connectionist Simulators
20 pages

RR-93-22
Manfred Meyer, Jörg Müller
Weak Looking-Ahead and its Application in Computer-Aided Process Planning
17 pages

RR-93-20
Franz Baader, Bernhard Hollunder
Embedding Defaults into Terminological Knowledge Representation Formalisms
34 pages

RR-93-18
Klaus Schild
Terminological Cycles and the Propositional μ-Calculus
32 pages

RR-93-17
Rolf Backofen
Regular Path Expressions in Feature Logic
37 pages

RR-93-16
Gert Smolka, Martin Henz, Jörg Würtz
Object-Oriented Concurrent Constraint Programming in Oz
17 pages

RR-93-15
Frank Berger, Thomas Fehrle, Kristof Klöckner, Volker Schölles, Markus A. Thies, Wolfgang Wahlster
PLUS - Plan-based User Support Final Project Report
33 pages

RR-93-14
Joachim Niehren, Andreas Podelski, Ralf Treinen
Equational and Membership Constraints for Infinite Trees
33 pages

RR-93-13
Franz Baader, Karl Schlechta
A Semantics for Open Normal Defaults via a Modified Preferential Approach
25 pages

RR-93-12
Pierre Sablayrolles
A Two-Level Semantics for French Expressions of Motion
51 pages

RR-93-11
Bernhard Nebel, Hans-Jürgen Bürckert
Reasoning about Temporal Relations: A Maximal Tractable Subclass of Allen's Interval Algebra
28 pages

RR-93-10
Martin Buchheit, Francesco M. Donini, Andrea Schaerf
Decidable Reasoning in Terminological Knowledge Representation Systems
35 pages

RR-93-09
Philipp Hanschke, Jörg Würtz
Satisfiability of the Smallest Binary Program
8 pages

RR-93-08
Harold Boley, Philipp Hanschke, Knut Hinkelmann, Manfred Meyer
CoLab: A Hybrid Knowledge Representation and Compilation Laboratory
64 pages

RR-93-07
Hans-Jürgen Bürckert, Bernhard Hollunder, Armin Laux
Concept Logics with Function Symbols
36 pages

RR-93-06
Hans-Jürgen Bürckert, Bernhard Hollunder, Armin Laux
On Skolemization in Constrained Logics
40 pages

RR-93-05
Franz Baader, Klaus Schulz
Combination Techniques and Decision Problems for Disunification
29 pages

RR-93-04
Christoph Klauck, Johannes Schwagereit
GGD: Graph Grammar Developer for features in CAD/CAM
13 pages

RR-93-03
Franz Baader, Berhard Hollunder, Bernhard Nebel, Hans-Jürgen Profitlich, Enrico Franconi
An Empirical Analysis of Optimization Techniques for Terminological Representation Systems
28 pages

RR-93-02
Wolfgang Wahlster, Elisabeth André, Wolfgang Finkler, Hans-Jürgen Profitlich, Thomas Rist
Plan-based Integration of Natural Language and Graphics Generation
50 pages

RR-93-01
Bernhard Hollunder
An Alternative Proof Method for Possibilistic Logic and its Application to Terminological Logics
25 pages

DFKI Technical Memos

TM-94-04
Cornelia Fischer
PAntUDE – An Anti-Unification Algorithm for Expressing Refined Generalizations
22 pages

TM-94-03
Victoria Hall
Uncertainty-Valued Horn Clauses
31 pages

TM-94-02
Rainer Bleisinger, Berthold Kröll
Representation of Non-Convex Time Intervals and Propagation of Non-Convex Relations
11 pages

TM-94-01
Rainer Bleisinger, Klaus-Peter Gores
Text Skimming as a Part in Paper Document Understanding
14 pages

TM-93-05
Michael Sintek
Indexing PROLOG Procedures into DAGs by Heuristic Classification
64 pages

TM-93-04
Hans-Günther Hein
Propagation Techniques in WAM-based Architectures — The FIDO-III Approach
105 pages

TM-93-03
Harold Boley, Ulrich Buhrmann, Christof Kremer
Konzeption einer deklarativen Wissensbasis über recyclingrelevante Materialien
11 pages

TM-93-02
Pierre Sablayrolles, Achim Schupeta
Conflict Resolving Negotiation for COoperative Schedule Management Agents (COSMA)
21 pages

TM-93-01
Otto Kühn, Andreas Birk
Reconstructive Integrated Explanation of Lathe Production Plans
20 pages

DFKI Documents

D-94-15
Stephan Oepen
German Nominal Syntax in HPSG
— On Syntactic Categories and Syntagmatic Relations —
80 pages

D-94-14
Hans-Ulrich Krieger, Ulrich Schaefer.
TDL - A Type Description Language for HPSG, Part 2: User Guide.
72 pages

D-94-12
Arthur Sehn, Serge Autexier (Hrsg.)
Proceedings des Studentenprogramms der 18. Deutschen Jahrestagung für Künstliche Intelligenz KI-94
69 Seiten

D-94-11
F. Baader, M. Buchheit, M. A. Jeusfeld, W. Nutt (Eds.)
Working Notes of the KI'94 Workshop: KRDB'94 - Reasoning about Structured Objects: Knowledge Representation Meets Databases
65 pages

Note: This document is no longer available in printed form.

D-94-10
F. Baader, M. Lenzerini, W. Nutt, P. F. Patel-Schneider (Eds.)
Working Notes of the 1994 International Workshop on Description Logics
118 pages

Note: This document is available for a nominal charge of 25 DM (or 15 US-$).

D-94-09
DFKI Wissenschaftlich-Technischer Jahresbericht 1993
145 Seiten

D-94-08
Harald Feibel
IGLOO 1.0 - Eine grafikunterstützte Beweisentwicklungsumgebung
58 Seiten

D-94-07
Claudia Wenzel, Rainer Hoch
Eine Übersicht über Information Retrieval (IR) und NLP-Verfahren zur Klassifikation von Texten
25 Seiten

D-94-06
Ulrich Buhrmann
Erstellung einer deklarativen Wissensbasis über recyclingrelevante Materialien
117 Seiten

D-94-04
Franz Schmalhofer, Ludger van Elst
Entwicklung von Expertensystemen: Prototypen, Tiefenmodellierung und kooperative Wissensevolution
22 Seiten

D-94-03
Franz Schmalhofer
Maschinelles Lernen: Eine kognitionswissenschaftliche Betrachtung
54 Seiten

D-94-02
Markus Steffens
Wissenserhebung und Analyse zum Entwicklungsprozeß eines Druckbehälters aus Faserverbundstoff
90 pages

D-94-01
Josua Boon (Ed.)
DFKI-Publications: The First Four Years
1990 - 1993
75 pages

D-93-27
Rolf Backofen, Hans-Ulrich Krieger, Stephen P. Spackman, Hans Uszkoreit (Eds.)
Report of the EAGLES Workshop on Implemented Formalisms at DFKI, Saarbrücken
110 pages

D-93-26
Frank Peters
Unterstützung des Experten bei der Formalisierung von Textwissen INFOCOM - Eine interaktive Formalisierungskomponente
58 Seiten

D-93-25
Hans-Jürgen Bürckert, Werner Nutt (Eds.)
Modeling Epistemic Propositions
118 pages

Note: This document is available for a nominal charge of 25 DM (or 15 US-$).

D-93-24
Brigitte Krenn, Martin Volk
DiTo-Datenbank: Datendokumentation zu Funktionsverbgefügen und Relativsätzen
66 Seiten

D-93-22
Andreas Abecker
Implementierung graphischer Benutzungsoberflächen mit Tcl/Tk und Common Lisp
44 Seiten

Note: This document is no longer available in printed form.

D-93-21
Dennis Drollinger
Intelligentes Backtracking in Inferenzsystemen am Beispiel Terminologischer Logiken
53 Seiten

D-93-20
Bernhard Herbig
Eine homogene Implementierungsebene für einen hybriden Wissensrepräsentationsformalismus
97 Seiten

D-93-16
Bernd Bachmann, Ansgar Bernardi, Christoph Klauck, Gabriele Schmidt
Design & KI
74 Seiten

D-93-15
Robert Laux
Untersuchung maschineller Lernverfahren und heuristischer Methoden im Hinblick auf deren Kombination zur Unterstützung eines Chart-Parsers
86 Seiten

D-93-14
Manfred Meyer (Ed.)
Constraint Processing – Proceedings of the International Workshop at CSAM'93, St.Petersburg, July 20-21, 1993
264 pages

Note: This document is available for a nominal charge of 25 DM (or 15 US-$).

D-93-12
Harold Boley, Klaus Elsbernd, Michael Herfert, Michael Sintek, Werner Stein
RELFUN Guide: Programming with Relations and Functions Made Easy
86 pages

D-93-11
Knut Hinkelmann, Armin Laux (Eds.)
DFKI Workshop on Knowledge Representation Techniques — Proceedings
88 pages

Note: This document is no longer available in printed form.

D-93-10
Elizabeth Hinkelman, Markus Vonerden, Christoph Jung
Natural Language Software Registry (Second Edition)
174 pages

D-93-09
Hans-Ulrich Krieger, Ulrich Schäfer
TDLExtraLight User's Guide
35 pages

D-93-08
Thomas Kieninger, Rainer Hoch
Ein Generator mit Anfragesystem für strukturierte Wörterbücher zur Unterstützung von Texterkennung und Textanalyse
125 Seiten

D-93-07
Klaus-Peter Gores, Rainer Bleisinger
Ein erwartungsgesteuerter Koordinator zur partiellen Textanalyse
53 Seiten

D-93-06
Jürgen Müller (Hrsg.)
Beiträge zum Gründungsworkshop der Fachgruppe Verteilte Künstliche Intelligenz, Saarbrücken, 29. - 30. April 1993
235 Seiten

Note: This document is available for a nominal charge of 25 DM (or 15 US-$).

D-93-05
Elisabeth André, Winfried Graf, Jochen Heinsohn, Bernhard Nebel, Hans-Jürgen Profitlich, Thomas Rist, Wolfgang Wahlster
PPP: Personalized Plan-Based Presenter
70 pages

D-93-04
DFKI Wissenschaftlich-Technischer Jahresbericht 1992
194 Seiten

D-93-03
Stephan Busemann, Karin Harbusch(Eds.)
DFKI Workshop on Natural Language Systems: Reusability and Modularity - Proceedings
74 pages

D-93-02
Gabriele Schmidt, Frank Peters, Gernod Laufkötter
User Manual of COKAM+
23 pages

D-93-01
Philipp Hanschke, Thom Frühwirth
Terminological Reasoning with Constraint Handling Rules
12 pages

Betty: Ein System zur Planung und Generierung informativer Animationssequenzen

D-95-02
Documen

Autor: Andreas Butz