JDraw ein Zeichenprogramm in Java

Im Rahmen der Veranstaltung Programmieren III

21. Januar 2001

Niels-Peter de Witt	Matrikelnr. 2083921
Karsten Wolke	Matrikelnr. 2083967
Helge Janicke	Matrikelnr. 2083973

Inhaltsverzeichnis

1 Vorwort

Dieses Projekt fand im Rahmen der Veranstaltung Programmieren III, an der FHOOW Standort Emden, statt. Ziel des Projektes ist den Umgang mit der Programmiersprache JAVA weiter zu vertiefen, sowie eine Dokumentation zu einem selbstgewählten Projekt zu schreiben.
Wir wählten uns als Projekt die Planung und Implementierung eines Zeichenprogramms, da es für uns eine Vielzahl neuer Einblicke in die Möglichkeiten von JAVA bieten konnte. Dabei legten wir wesentlich mehr Gewicht auf den Entwurf der Logik und Datenstrukturen, als auf die Implementierung und die Gestaltung der grafischen Oberfläche.
Implementiert wurden von uns nur kleine Beispiele, die das Konzept verdeutlichen und die als Vorlagen für weitere Implementierungen dienen.
Danken wollen wir an dieser Stelle noch Herrn Prof. Siemsen, der diese Veranstaltung betreute und Herrn Prof. Totzauer, der mit der Veranstaltung Algorithmen und Datenstrukturen die Grundsteine für die verwendeten Datenstrukturen legte. Wir hoffen, daß die folgenden Seiten einen brauchbaren Eindruck von dem Konzept des Zeichenprogrammes geben können.

Helge Janicke, Niels Peter de Witt, Karsten Wolke

2 Überblick

Das gesamte Projekt läßt sich in 3 Teile unterteilen:

- Aufbau einer Zeichnung - die Datenstruktur.
- Benutzerinteraktion - das Eventhandling.
- Grafische Oberfläche - GUI.

In den folgenden Kapiteln wollen wir näher auf diese Punkte eingehen, die Probleme und unsere Lösungen darstellen. Wir hoffen, daß der aufmerksame Leser die Probleme und die sich ergebenen Lösungswege nachvollziehen kann und vielleicht die ein oder andere Verbesserung findet. Für Anmerkungen und neue Ideen sind wir offen und auch dankbar! Denn obwohl das Projekt mit diesem Bericht abgeschlossen ist werden wir (wenn Zeit dazu ist) das Programm weiterentwickeln.
Falls Sie Interressee an diesem Projekt finden sollten, informieren wir Sie gerne über den aktuellen Stand und die zukünftige Planung.

Wenden Sie sich in diesem Fall bitte an:

Helge Janicke:
email: heljanic@hermes.fho-emden.de

Niels Peter de Witt:
email: niewitt@hermes.fho-emden.de

Karsten Wolke:
email: mail@karsten-wolke.de

Doch nun zum weiter zum ersten Teil....

3 Aufbau einer Zeichnung - die Datenstruktur

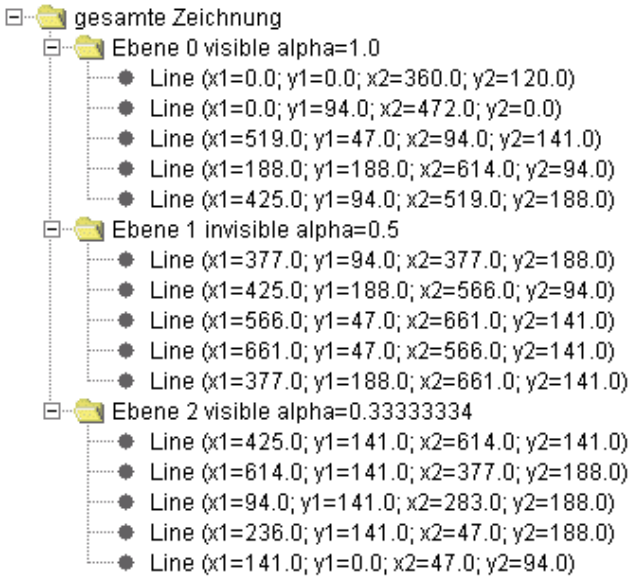
Am Anfang stand für uns die Überlegung was wir in denn alles in dieser Datenstruktur alles speichern wollten, also was für Informationen notwendig sind um eine Zeichnung aufzubauen. So erhielten wir folgende Liste:

- Strukturinformationen über:
  - Ebenen
    Das Zeichnen in mehreren Ebenen soll möglich sein. Die einzelnen Ebenen sollen einen eigenen Alphakanal haben (Alpha-Blending).
  - Gruppen
    Es ist sinnvoll Gruppen bilden zu können, so das einmal gezeichnete komplexere Objekte in einem festen Zusammenhang stehen.
  - Bildgröße
    Natürlich müssen auch Informationen über das Bild an sich gespeichert werden.
- Darstellbare Objekte mit folgenden Eigenschaften:
  - Position
    Die Position eines Zeichen-Objektes muß im nachhinein veränderbar sein.

- Größe
  Die Größe muß sich im nachhinein anpassen lassen.
- Farbe
- Linie
- Füllung
- etc.

Mit diesen Anforderungen erschien es uns am sinnvollsten eine Baumstruktur zu wählen, da diese die Struktur eines Bildes mit mehreren Ebenen am besten wiedergibt.

Screenshot der Baumansicht einer Zeichnung



Der Wurzelknoten entspricht dem Gesamtbild. Die Knoten mit der Höhe 1 entsprechen den einzelnen Ebenen. Alle weiteren Knoten sind Gruppierungen, während die Blätter des Baumes die einzelnen Zeichenobjekte wie zum Beispiel Linien, Rechtecke und Kreise darstellen.

Die einzelnen Objekte wie das Gesamtbild, die Ebene, die Gruppe und natürlich auch ein Zeichenobjekt werden durch eigene Klassen repräsentiert, auf die wir im nächsten Abschnitt näher eingehen werden.

4 Die Klassen und Interfaces, die Bildelemente repräsentieren

4.1 Das Gesamtbild:

Für die Informationen über die Bildgröße und andere Eigenschaften, wie z.B. Author etc. gibt es noch keine Klasse, da diese Informationen für die grundlegende Struktur nicht notwendig sind. Die Bildgröße wurde für Testzwecke auf einen festen Wert in Punkten gesetzt (weißer Bereich in der Zeichnung).

4.2 Eine Ebene: Layer.java

Auf einer Ebene können sich Gruppierungen und einzelne Zeichenobjekte befinden. Die Ebene fasst diese Objekte nocheinmal zusammen zu einer logischen Gruppe. Für eine Ebene kann ein Alphawert gesetzt werden, so das einzelne Ebenen durchsichtig erscheinen können. Layer.java ist eine Klasse die Informationen über den Alphakanal enthält und ein flag, ob diese Ebene gerade die aktive Ebene ist. Es ist dem Benutzer immer nur möglich auf einer Ebene zu zeichnen, auf der gerade aktiven. Es ist vorgesehen

Zeichenobjekte und Gruppen zwischen den Ebenen austauschen zu können. Auch ein Zusammenfassen von Ebenen wird möglich sein.

4.3 Eine Gruppe: AbstractGraphGroup.java

Eine Klasse für eine Gruppierung ist im Momment noch nicht implementiert. Es ist also noch nicht möglich dauerhafte Gruppierungen zu erzeugen. Es existiert jedoch schon eine spezielle Form einer Gruppierung, die von uns benutzt wird um Mehrfachselektionen gemeinsam zu bearbeiten. Eine Gruppe muß die einzelnen Zeichenobjekte und Gruppen kapseln und Transformationen an alle Gruppenmitglieder weitergeben.

4.4 Ein Zeichenobjekt: AbstractGraphObject.java

Ein Zeichenobjekt ist eine Linie, Kreis, Rechteck oder allgemein ein Polygonzug. Ein solches Objekt kann verschieden Eigenschaften, wie Farbe, Linienstil etc. haben. Die Informationen über Form und Aussehen werden in einer speziellen Klasse gespeichert, wie zum Beispiel jdLine.java. Alle Klassen, die ein Zeichenobjekt repräsentieren müssen eine Unterklasse von AbstractGraphObject sein. AbstractGraphObject stellt insbesondere eine Methode zur Verfügung, die prüft, ob ein Rechteck dieses Zeichenobjekt schneidet. Dies ist besonders im Bezug auf die Auswahl von Objekten von Wichtigkeit. Ein Zeichenobjekt kann allerdings noch andere Eigenschaften besitzen, die über Interfaces deklariert werden. Einige Beispiele sind:

- Selectable.java
  Dieses Interface gibt an, daß das Objekt, nach dem es gezeichnet wurde auch wieder ausgewählt werden kann. Dieses Interface wird auch von Gruppen implementiert.

- Transformable.java
  Dieses Interface spezifiziert Methoden, über die ein Zeichenobjekt transformiert, also gedreht, verschoben und gestreckt werden kann. Dies ist wohl eines der komplexesten Interfaces. Transformationen werden über eine Affine Transformation durchgeführt. Jedes Zeichenobjekt besitzt eine eigene Transformationsmatrix, die angewendet wird, bevor das Objekt gezeichnet wird.

- Infoable.java
  Ein Objekt das dieses Interface implementiert ist in der Lage die Eigenschaften des Objektes in einem Panel darzustellen. Dieses Panel wird vom Zeichenprogram unterhalb der Zeichenfläche dargestellt. Es ist prizipiell auch möglich dort Eingaben zum ändern von Eigenschaften zu machen.

- Drawable.java.
  Drawable spezifiziert, wie ein Objekt vom Anwender gezeichnet wird. Eine Klasse die dieses Interface implementiert ist also in der Lage MouseEvents solange auszuwerten, bis ein vollständiges Zeichenobjekt entstanden ist. Dies ermöglicht die optimale Erweiterbarkeit des Zeichenprogrammes. Soll ein neues Zeichengerät in das bestehende System eingebunden werden, so sind dazu zwei Klassen notwendig. Eine Klasse repräsentiert das Zeichenobjekt selbst, die andere die Bauanleitung für dieses Objekt. Sobald die Bauanleitung das Interface Drawable erfüllt kann ein solches Zeichenobjekt von unserem Zeichenprogramm dargestellt werden. Mit diesem Konzept ist es auch denkbar Zeichentools als Plug-Ins zur Verfügung zu stellen.

Bisher wurden von uns nur eine beispielhafte Implementationen von Linien und eine unvollständige Implementation von Rechtecken entwickelt. Anhand der dort verwendeten Konzepte sollte es allerdings leicht sein auch andere Zeichentools zu entwickeln.

5 Die Darstellung des Bildes

Die Darstellung der Bildes unterteilt sich in zwei verschiedene Bereiche.
Erstens die Darstellung der aktuellen im Baum befindlichen Zeichenobjekte und zweitens die Darstellung der vom Benutzer selektierten Objekte. Um die Auswahl und den Datenbestand zu trennen entwickelten wir zwei Klassen, die in der Lage sind Zeichenobjekte darzustellen.
Das Gesamtbild ergibt sich dann aus der Darstellung des Baumes und einer (mit transparentem Hintergrund) überlagerten Darstellung der Auswahl.

5.1 Darstellung des Baumes: LayerPainter.java

Diese Klasse kann von einem Knoten im Baum ausgehend alle Zeichenobjekte vom Typ AbstractGraphObject darstellen. Dazu wird der Unterbaum in Preorder-Reihenfolge durchlaufen. Dies bedeutet, daß Objekte die am Anfang gezeichnet wurden bei der Darstellung auch wieder zuerst gezeichnet werden. Ist das zu zeichnende Objekt eine Ebene, so wird das Alpha der Ebene vor dem Zeichnen gesetzt (Es ist auch denkbar ein Alphawert für jedes Objekt zu definieren). Vor dem Zeichnen eines Zeichenobjektes (z.B. Linie) wird die Farbe, der Linienstil und die Füllart, bzw. Farbe gesetzt, und das Zeichenobjekt gemäß seiner Transformationsmatrix transformiert.

5.2 Darstellung der Auswahl: GlassPane.java

Das Interface GlassPane.java definiert alle Methoden, die notwendig sind eine Liste von Zeichenobjekten darzustellen. In einer einfachen Implementation (DefaultGlassPane.java) wird zusätzlich noch die Verwaltung der Infoable-Objekte (siehe voriges Kapitel) übernommen. Die GlassPane verwaltet also die gesamte Auswahl. Werden mehrere Objekte gleichzeitig selektiert, so erstellt die GlassPane einen losen Zusammenhang - ein ImplicitGroup Objekt. Auf diese Weise kann das Zeichenprogramm mehrfache Auswahlen synchron bearbeiten.

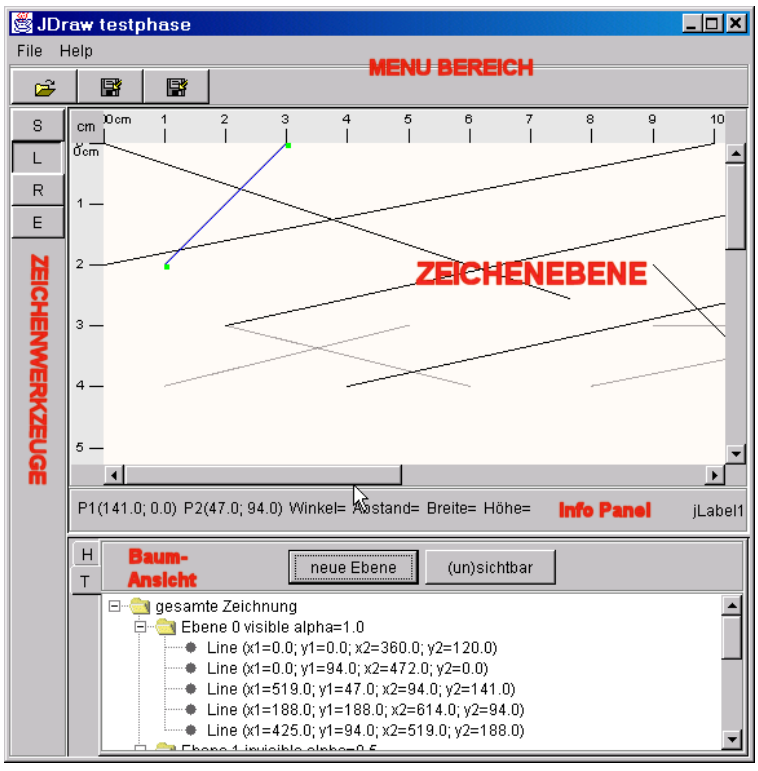
6 Die Interaktion des Benutzers

Die Eingaben des Benutzers werden in zwei grundlegende Teile getrennt, den Teil, in dem Einstellungen und Zeichenwerkzeuge gewählt werden können (dem Hauptprogramm MainFrame.java) und einem Teil, der die tatsächlichen Zeichenvorgänge behandelt (dem allgemeinen Listener UserDrawAction.java).

6.1 Das Hauptprogramm MainFrame.java

Das Hauptprogramm dient hauptsächlich dazu die Verwaltungsaufgaben, und Menüs zu verwalten. Hier werden die einzelnen gekapselten Teilbereiche eingebunden. Es besteht ein weitreichender Zugriff auf die meisten Daten. Zur Beschreibung der äußeren Form ein Screenshot:

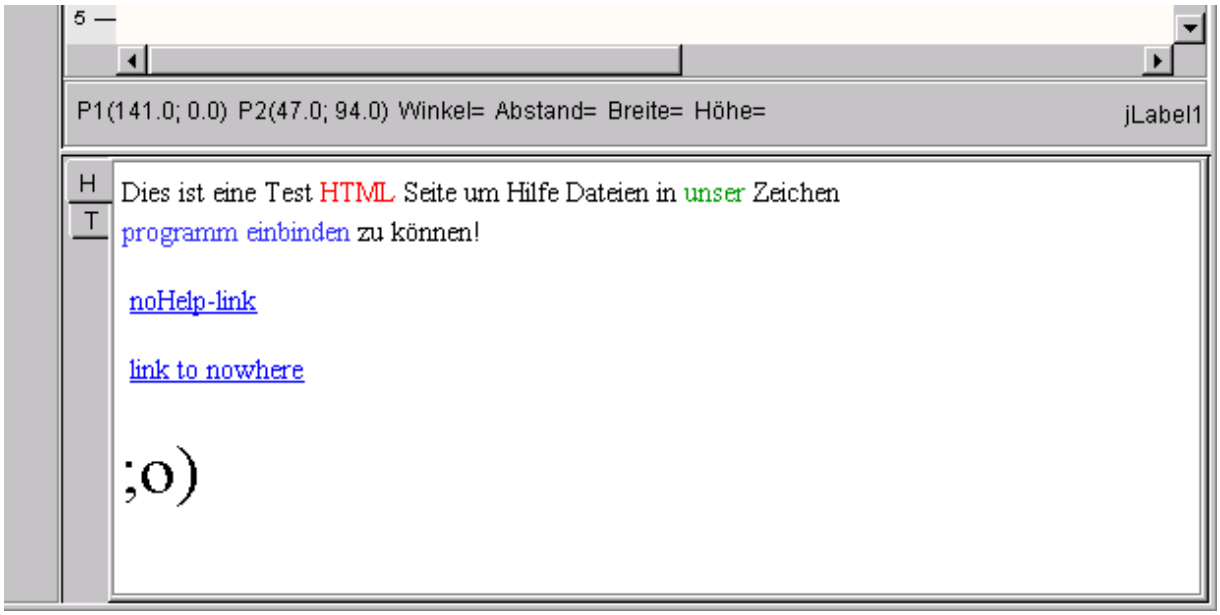
Screenshot des Zeichenprogrammes in der aktuellen Version.



6.2 Die Ereignisbehandlung beim Zeichnen

Bei der Behandlung von Zeichenereignissen kommt das Konzept der plugin-fähigen Zeichengeräte zum Tragen. Die Klasse, die die Ereignisse zum Zeichnen verarbeitet reagiert in der jetzigen Version nur auf Mouseereignisse (Sie ist eine sogenannte MouseAdapter-Klasse). Es ist jedoch geplant diese Klasse mit Methoden zum Reagieren auf Tastaturereignisse zu erweitern. Auch würde sich hier die Möglichkeit bieten ein grafisches Tablett als Eingabegerät anzusteuern.
Die Struktur in der Mousevents gehandhabt werden ist relativ einfach. Es gibt die Unterscheidung zwischen einem Auswahlmodus, und einem Zeichenmodus (es sind noch weitere Modi für Rotation und Scherung vorgesehen). Befindet sich der Benutzer im Auswahlmodus so können bereits vorhandene Zeichenobjekte verschoben, selektiert, und über ihre AccessPoints (Punkte zum "anfassen" der Objekte und ändern von Objekteigenschaften, aber dazu später mehr) manipuliert werden. Befindet sich der Benutzer im Zeichenmodus, so gibt es ein ausgewähltes Zeichengerät, welches durch eine Klasse vom Typ Drawable repräsentiert wird. Diese Klasse Drawable enthält Methoden, die auf Mouseereignisse reagieren und Objekte, die gerade gezeichnet werden, darstellen. Dazu erhält die Drawable Klasse Zugriff auf die Glasspane, um dort temporär Informationen anzeigen zu können. Es ist Konvention das sobald eine dieser Methoden den boolschen Wert true zurückliefert das zu zeichnende Objekt als abgeschlossen gilt und von der Drawable Klasse eine Instanz des gerade gezeichneten Objektes angefordert werden kann. Dieses Objekt wird dann in das Bild gezeichnet - also in den Baum eingefügt, als das älteste Kind in der aktuellen Ebene. Zu den jeweiligen Zeichengeräten gibt es eine kontextbezogene Hilfe, die auch von der Klasse Drawable bereitgestellt werden kann. Sie erscheint in einer Registerkarte dort wo sich auch die Baumansicht der aktuellen Zeichnung befindet. Dieser Hilfetext ist im HTML Format und kann auch Links zu anderen lokalen Seite, oder wenn eine Verbindung zum Internet besteht, auch auf im Netz liegende Seiten enthalten. Hier ein Screenshot:

Screenshot der kontextbezogenen Hilfe.



7 Nur kurz angeschnitten...

7.1 Ändern bestehender Objekte

7.1.1 Access-Points

Um bereits in der Zeichnung vorhandene Objekte zu verändern bedienen wir uns sogennanter Access-Points. Access-Points sind kleine Bereiche, die für ein Zeichenobjekt markante Stellen hervorheben - zum Beispiel bei einer Linie die beiden Endpunkte. Diese Access-Points können nun angeklickt und via Drag and Drop verschoben werden. Somit ist es möglich ein bestehendes Zeichenobjekt im Nachhinein

zu verändern. Die Access-Points werden durch die abstrakte Klasse AbstractAccessPoint.java definiert. Eine einfache Implementation ist die Klasse SimpleAccessPoint.java. Hiermit können verschiedene Typen von Access-Points realisiert werden, zum Beispiel ein Access-Point, den man nur in Richtung Nord-West verschieben kann, oder einen der für die Drehung des Zeichenobjektes zuständig ist.
Ein Zeichenobjekt, das durch eine Klasse repräsentiert wird, die das Transformable Interface implementiert, muß auch Methoden bereitstellen, welche Access-Points für dieses Objekt zur Verfügung stellen können. Auch werden Ereignisse die diese Access-Points betreffen an diese Klasse weitergeleitet (Vergleiche Drawable.java Interface). Dadurch ist es möglich Access-Points für jeden erdenklichen Zweck zur Verfügung zu stellen, der sich für das Zeichenobjekt eignet.

7.1.2 Zugriff über die Baumstruktur

Es ist geplant die Eigenschaften eines Zeichenobjektes auch über Tastatureingaben zu ändern. Dazu muß allerdings ein Zeichengerät ein Dialogfenster zur Verfügung stellen, in der die jeweiligen Objekteigenschaften geändert werden können. Dies wird über ein weiteres Interface realisiert werden. Auch das Löschen von Zeichenobjekten, Gruppen oder Ebenen, sowie das Kopieren und Einfügen von Objekten soll über den Baum möglich sein. Um zu Zeigen das dies prinzipiell möglich ist haben wir das Löschen, sowie die Möglichkeit Objekte im Baum zu selektieren implementiert.

7.2 Maßeinheiten und Fangfunktionen

Um die Möglichkeit zu schaffen mit verschiedenen Maßeinheiten zu arbeiten haben wir eine Klasse Unit.java implementiert, die Methoden zum Umrechnen von einer Maßeinheit in eine andere bereitstellt. Diese Klasse wird auch verwendet, um ein Lineal an den Rändern in einer bestimmten Maßeinheit anzuzeigen. Intern werden die Zeichenobjekte allerdings in Punkten gespeichert.
Auch Fangfunktionen können über mit diesen Maßeinheiten gesteuert werden. Allerdings besteht hierbei noch die Möglichkeit feinere Unterteilungen einzustellen - in Anteilen der Einheit. Als einfache Fangfunktion haben wir ein Gitter implementiert. Diese Klassen sind bisher nur teilweise fertiggestellt, bieten aber schon die Möglichkeit beim Zeichnen eines neuen Objektes die Punkte an dem Gitter auszurichten. Die graphische Darstellung eines Gitters ist abstrakt definiert aber noch nicht umgesetzt. Maßeinheiten werden über einen Knopf in der linken oberen Ecke des Zeichenfeldes umgeschaltet.

8 Ausblick auf die weitere Entwicklung

Das dieses Projekt in der kurzen Zeit (zwei Semesterwochenstunden) nicht fertiggestellt werden kann war schon bei Beginn klar. Das wir trotzdem viel Zeit in ein vernünftiges Grundkonzept stecken und damit weitere Verzögerungen in Kauf nahmen können wir auch verantworten, denn wann hat man sonst je nocheinmal die Gelegenheit sich intensiv und ohne großen Zeitdruck mit der Planung eines Projektes zu beschäftigen - wenn nicht im Studium. Ein weiterer Gesichtspunkt war das wir durchaus vorhaben dieses Projekt - wie Eingangs erwähnt - weiterzuführen. Somit kommt uns auch ein gutes Konzept zu Gute - denn alle Erweiterungen, die wir bis jetzt angedacht haben ließen sich in sehr kurzer Zeit als Prototyp realisieren. Was noch alles von der jetzigen Struktur realisierbar und auch geplant ist wollen wir hier kurz auflisten.

- Implementierung weiterer grundlegender Zeichenfunktionen:
  - Rechtecke
  - Ellipsen
  - Polygonzüge
  - Text
- Kopieren, Löschen, Einfügen.
- Speichern der Dateien:
  - In einem eigenen Zeichenformat.
  - Als Postscript.

  - Als jpeg oder gif.

- Einstellen der Bildinformationen wie Größe etc.
- Ändern von Farbe, Linienstil und Füllung.
- Einbinden von Bitmaps (gif, jpeg, png, bmp).
- Rotation und Scherung von Zeichenobjekten.
- Gruppieren von Objekten.
- Erweiterung des Alphablending auf einzelne Objekte und Verfeinerung auf die verschiedenen Anwendungsarten des Alphablending.
- Fanglinien.
- Lupenfunktion.
- Druckfunktion.

Wir hoffen, daß wir die Zeit finden all diese für ein Zeichenprogramm notwendigen Funktionen umsetzen zu können.