VL8: Von ASCII nach Postscript

(Noch nicht fertig)

Inhalt

1. Einleitung

Im bisherigen Verlauf der Lehrveranstaltung wurde in stark vereinfachter Form verdeutlicht, was unter einem Softwareprojekt im Rahmen des MDA-Ansatzes der OMG zu verstehen ist. Das allgemeine Konzept wurde am Beispiel einer konkreten Problemstellung konkretisiert und illustriert. In diesem Zusammenhang sind auch erste Grundelemente der Programmiersprache C++ eingeführt worden. Bevor nun im Rahmen des Anwendungsbeispieles weitere Elemente der Programmiersprache C++ erläutert werden, soll in der heutigen Vorlesung der Übergang von einem ASCII-Text zu einem Postscript-Programm etwas eingehender erläutert werden. Als Referenz für die Sprache Postscript benutzen wir nicht die allerneueste Version 3.0 der Sprache, sondern die Versionen 1.5 sowie 2.0. Dies wird damit begründet, dass eine Einführung über diese früheren Versionen einfacher ist; die späteren Neuerungen lassen sich vor diesem Hintergrund dann leicht verstehen. Für einen Zugang zu den Postscript-Spezifikationen sei auf das Literaturverzeichnis http://www.fbmnd.fh-frankfurt.de/~doeben/II-PPmP/ii-ppmp05-header.html verwiesen.

2.Postscript-Programme

Im folgenden Schaubild (Bild 1) wird gezeigt, wie der grundsätzliche Zusammenhang zwischen ASCII-Texten und Postscript-Programmen in unserem Anwendungsbeispiel ist.

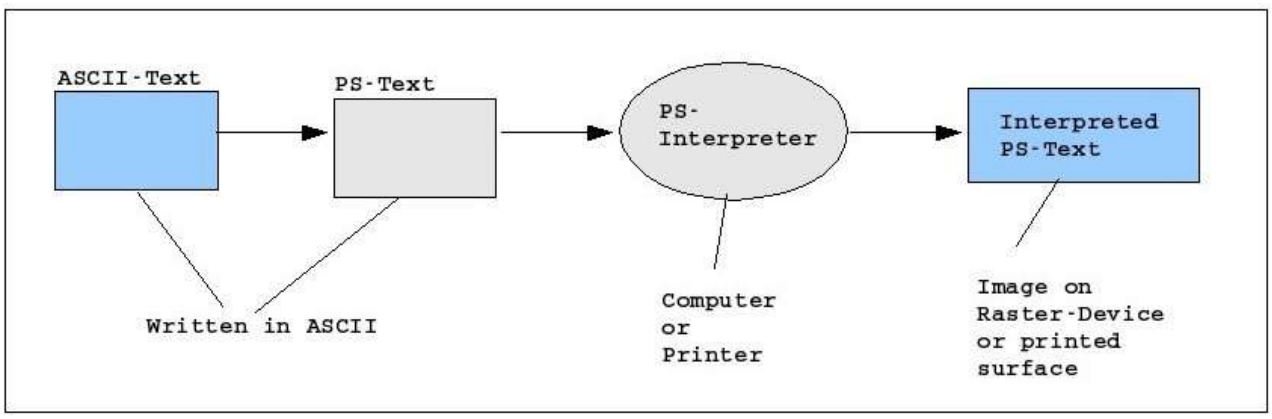

Bild 1: Vom ASCII-Text zum Postscript-Programm

Ein *Text im ASCII-Format,* der einen bestimmten Anwendungsfall repräsentiert, wird in einen *ASCII-Text* übersetzt, der den Regeln eines *Postscript-Programms* Version 1.5 (oder höher) entspricht. Ein solches Postscript-Programm kann von einem *Postscript-Interpreter* nach den Regeln eines Postscript-Interpreters *visualisiert* werden. Das Ergebnis einer solchen Interpretation wird entweder auf dem *Bildschirm* eines Computers ausgegeben oder als eine *bedruckte Papierseite* als Ergebnis eines Druckvorganges.

Ein Postscript-Programm ist also selbst wieder ein 7-Bit-ASCII-Text, der 128 Zeichen umfassen kann. Gross-/Kleinschreibung wird unterschieden. Bei Fliesskommazahlen wird der Punkt als Dezimalpunkt benutzt. Kommentarzeilen werden durch das Prozentzeichen ('%') am Anfang der Zeile gekennzeichnet.

Ein Postscript-Programm beschreibt letztlich einen Prozess, durch den eine Zeichnung auf einem Blatt Papier erstellt wird. Im Falle des DIN A4-Formates wird angenommen, dass ein Blatt eine Breite von 21cm und eine Höhe von 29,7cm besitzt. Diese Abmessungen werden auf ein Raster abgebildt, das dem Punktsystem des Pica-Systems entspricht. 1 Punkt (Pixel) ist der 72ste Teil eines amerikanischen Inches (1 Inch = 1 Zoll = 2,54cm; 2,54cm/72 = 1 Punkt = 0,03527778 cm, also ca. 0,0353 cm = 0,353 mm). (Zu Fragen von Schriftdarstellungen siehe z.B.: http://www.fabrice-pascal.de/artikel/sizediscussion/; zu den DIN-Formaten siehe: http://www.tanczos.at/Seiten/Informationen/Din-Formate.html).

Innerhalb des Interpreters existiert also eine Rasterdarstellung der DINA A 4 Seite, wie sie im Bild 2 veranschaulicht wird. Der Ursprungspunkt des x-y-Koordinatensystems liegt links unten (0,0). Die rechte obere Ecke einer DIN A 4 Seite würde bei (x=595,28, y=841,89) liegen. Bei realen Druckern muss man allerdings berücksichtigen, dass diese theoretischen Werte nie ganz erreicht werden, da immer ein kleiner Rand nichtdruckbarer Pixel existiert. Theoretisch könnte man auch Punkte jenseits der DINA A 4 Bereiche benutzen, also auch negative Koordinaten, aber diese würden bei einer DIN A 4- Darstellung dann unsichtbar bleiben.

Hier ein einfaches Beispiel, das Postscript-Programm *bild1.ps*.

```
% program bild1.ps
%
% simple mountain


0 0 moveto
290 310 lineto
595 0 lineto
closepath
stroke
showpage
```

Das Programm beginnt damit, dass ein Startpunkt (0,0) mit dem Befehl moveto gesetzt wird. Man sieht, dass Postscript die Postfix-Schreibweise für Befehle benutzt: erst die Argumente --hier die Koordinatenwerte 0 und 0 für x bzw. y--, dann das eigentliche Befehlswort *moveto*. Der befehl *0 0 moveto* setzt den Zeichenstift auf den Punkt (0,0). Mit dem Befehl *290 310 lineto* wird dann ein weiterer Punkt bei (290,319) gesetzt und zwischen den letzten beiden Punkten eine Linie gezogen. Entsprechend mit dem Befehl *595 0 lineto*. Dann wird mit dem Befehl *closepath* der Zeichenpfad geschlossen, mit dem Befehl *stroke* soll dann die Zeichnung ausgeführt und mit *showpage* sichtbar gemacht werden.

Unter Linux kann man das Programm mit dem Befehl *gv bild1.ps* auf der Konsole aufrufen und interpretieren lassen (siehe das Ergebnis im Bild 3).

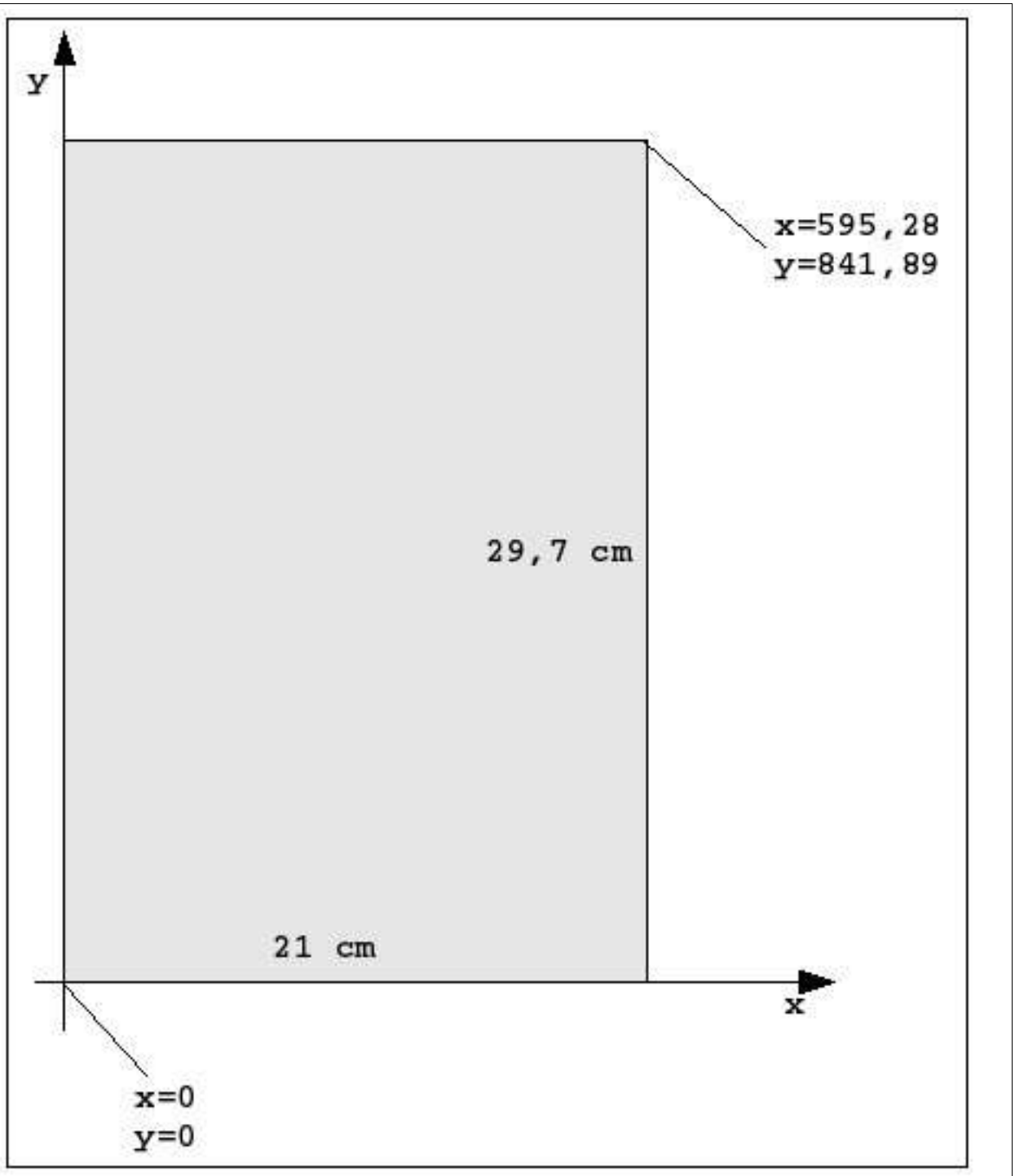

Bild 2: Eckwerte für eine DIN A 4 Seite mit cm und Pixel per Inch

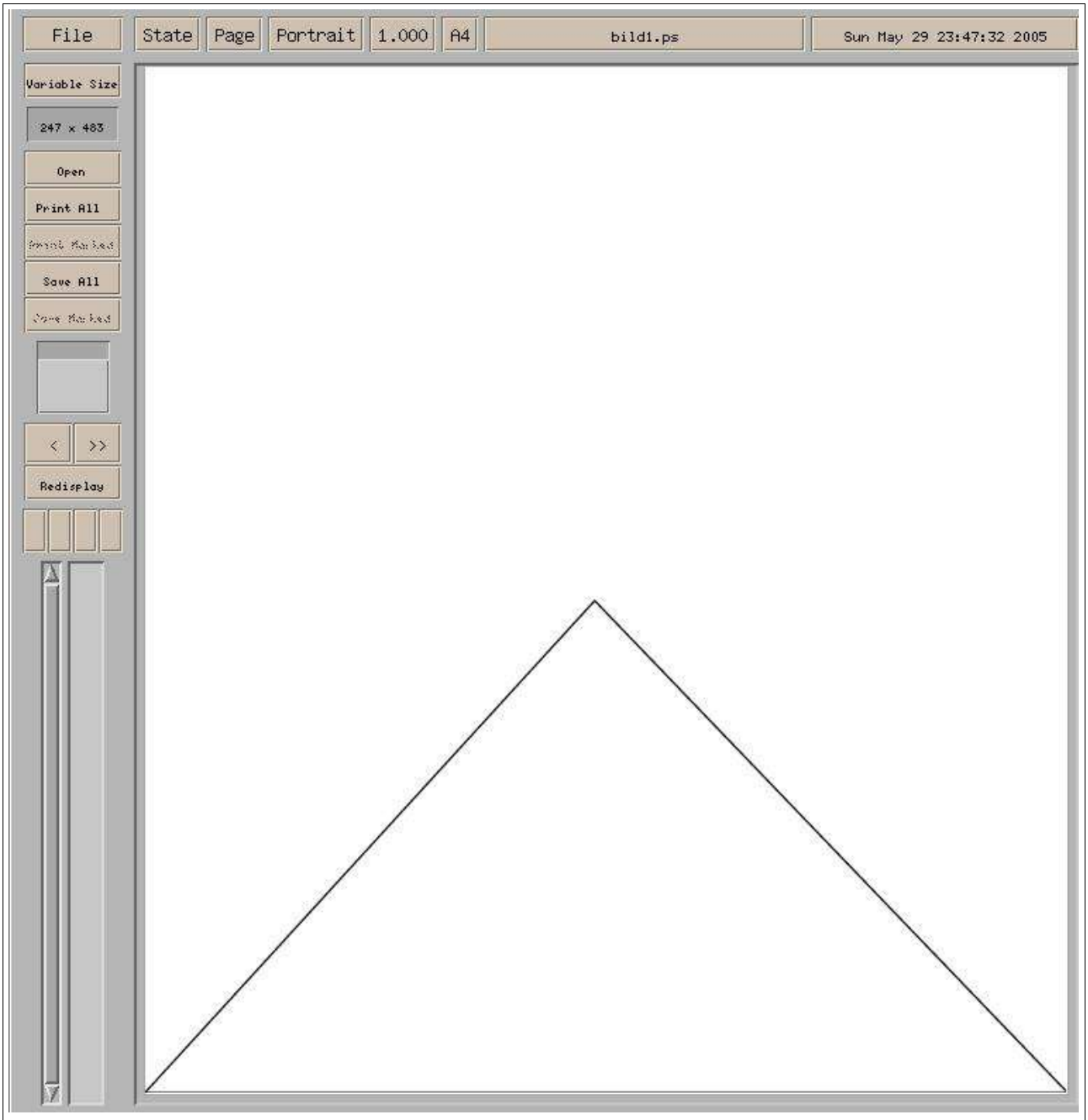

Bild 3: Visualisierung des Postscript-Programms bild1.ps mit gv

Im Programm bild2.ps wird ein Kreissegment hinzugefügt. Dies geschieht mit dem Befehl *x y r w1 w2 arc*, wobei (x,y) den Kreismittelpunkt bezeichnet, r den Radius, w1 den startwinkel, w2 den Endwinkel und der Kreisbogen von w1 bis w2 gezogen wird. w1=0 bedeuted in einem x-y-Koordinatensystem, dass der Punkt auf der x-Achse liegt und der Kreisbogen gegen den Uhrzeigersinn gezeichnet wird. w1=0 und w2=360 bedeutet einen vollständigen Kreisbogen.

```
% program bild2.ps
%
% simple mountain with simple sun


0 0 moveto
290 310 lineto
595 0 lineto
290 310 moveto
290 310 15 316 230 arc
closepath
stroke
showpage
```

Wichtig in diesem Beispiel ist das zweite moveto, damit der Endpunkt der letzten Zeichenoperation mit dem Mittelpunkt des Kreises zusammenfällt. Lässt man dieses moveto aus (probieren sie es), dann würde eine zusätzliche Linie vom Punkt (595,0) zum Punkt (290,310) gezogen.

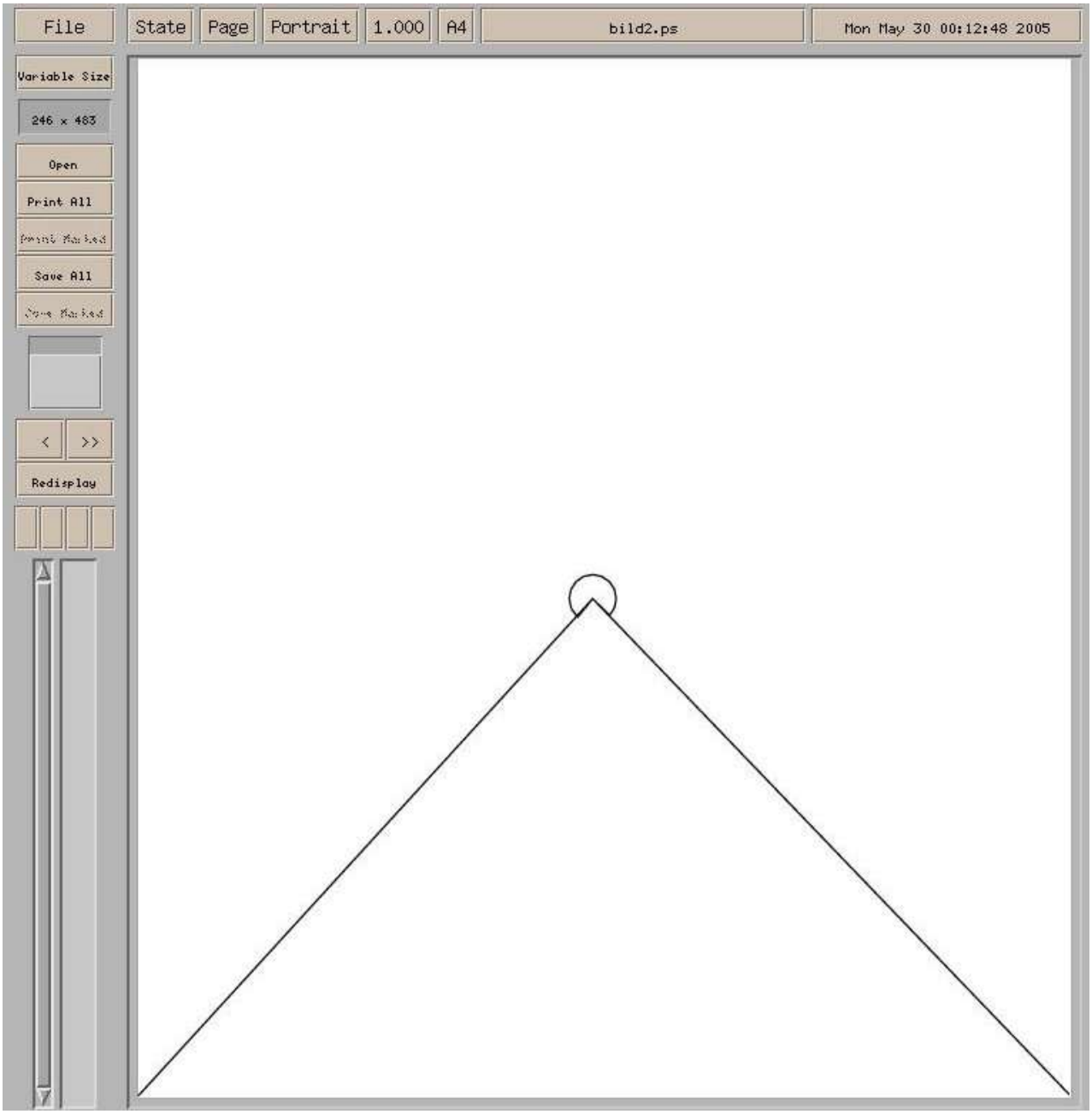

Bild 4: Postscriptdatei bild2.ps visualisiert mit gv

Im nachfolgenden Programm bild3.ps wird mit dem Befehl *N setgray* der Farbton gesetzt. Benutzt man nur Grautöne --wie im Beispiel--, dann liegen die Werte zwischen 0.0 (= schwarz) bis 1.0 (=weiss). Der Befehl *fill* sagt dann, dass die Fläche des letzten *closepath* ausgefüllt werden soll. Dabei wird in diesem Kontext deutlich, dass der Befehl *closepath* eine bislang verborgene Eigenschaft besitzt: *closepath* verbindet nämlich den Endpunkt des letzten Zeichenbefehls auch mit dem Anfangspunkt, so dass eine geschlossene Fläche entsteht. Ohne *closepath* würde diese Verbindung nicht erstellt. Ferner wird im Programm der Befehl *N setlinewidth* benutzt. Damit kann man die Linienstärke setzen. N=1 ist ein Punkt, N=3 sind 3 Punkte (Pixel).

```
% program bild3.ps
%
% simple mountain with simple sun

0 0 moveto
290 310 lineto
595 0 lineto
closepath

% Set color new
0.3 setgray
fill

290 310 moveto
290 310 15 316 230 arc
closepath

% set linestroke
3 setlinewidth
% Set color new
0.0 setgray
stroke

showpage
```

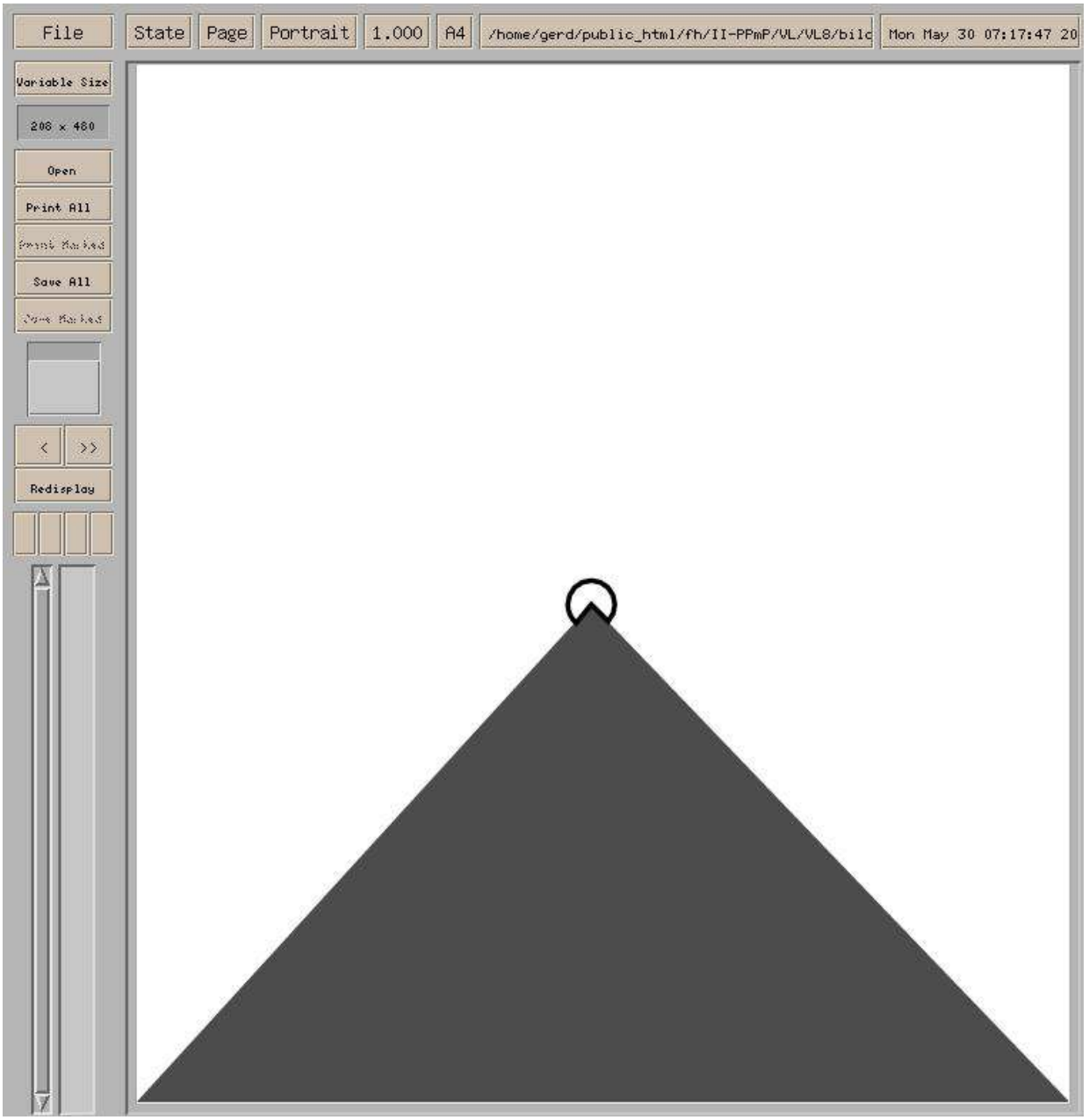

Bild 5: Füllen einer Fläche und Setzen des Farbtones

Im Programm bild4.ps tritt ein weiteres neues Element hinzu: Text. Mit dem Befehl

/Courier findfont wird dem Interpreter gesagt, er soll den Zeichensatz *Courier* finden und laden. Mit dem Befehl *N scalefont* wird die Grösse in Punkten (Pixel) festgelegt. *moveto* gibt dann den Punkt ab, ab wo im Bild der Text im Bild von links nach rechts geschrieben werden soll. Mit *(It's Sunrise) show* schliesslich wird gesagt, dass der Text zwischen den runden Klammern gedruckt werden soll (siehe Bild 6).

```
% program bild4.ps
%
% simple mountain with simple sun

0 0 moveto
290 310 lineto
595 0 lineto
closepath
% Set color new
0.3 setgray
fill

290 310 moveto
290 310 15 316 230 arc
closepath
% set linestroke
3 setlinewidth
% Set color new
0.0 setgray
stroke

% With text

/Courier findfont
24 scalefont
setfont
150 410 moveto
(It's Sunrise) show

showpage
```

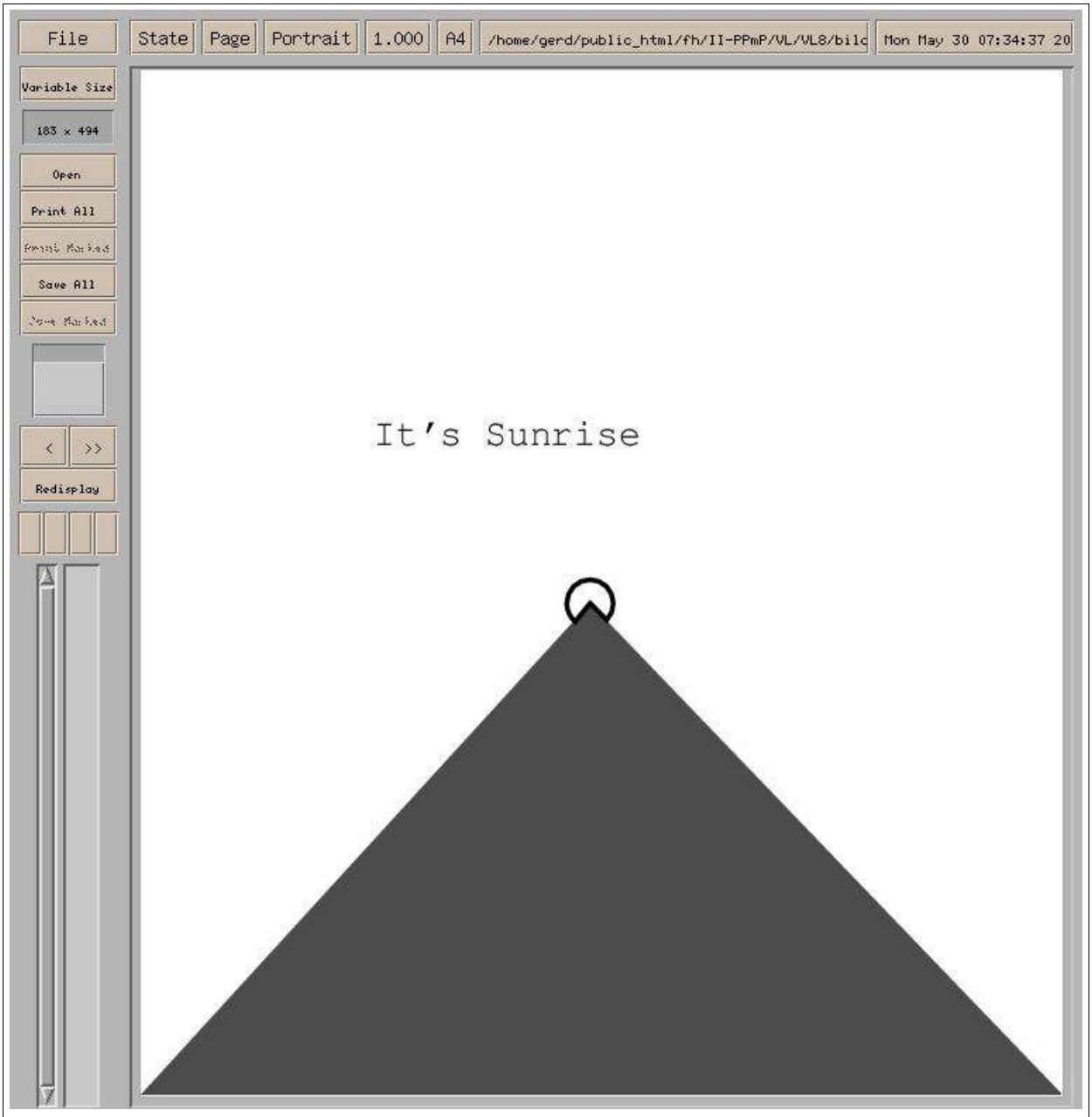

Bild 6: Einfügen eines Textes

3. Von ASCII nach PS

Nachdem nun geklärt ist, wie man einfache Postscript-Programme erstellen kann, soll nun nochmals die Aufgabenstellung weitergehender präzisiert werden.

Allgemein soll ja ein ASCII-Text in ein PS-Programm übersetzt werden. Dies macht natürlich nur Sinn, wenn der ASCII-Text Sachverhalte beschreibt, die sich als PS-Programme darstellen lassen. Und da PS-Programme *Zeichnungen* darstellen, sind also nur solche Sachverhalte interessant, die sich *zeichnen* lassen.

Das folgende einfache Postscript-Programm *bild7.ps* zeigt eine Strasse (angedeutet durch zwei Linien) mit einem auto (graues Rechteck mit Markierung an der Vorderseite und Beschriftung). Ein Text, der

```
% program bild5.ps
%
% street with car

% Drawing a road

10 10 moveto
400 10 lineto
closepath
10 210 moveto
400 210 lineto
closepath
3 setlinewidth
0.0 setgray
stroke

% Drawing a Car 1

40 20 moveto
40 70 lineto
140 70 lineto
140 20 lineto
closepath
0.7 setgray
fill
130 35 moveto
130 55 lineto
140 45 lineto
closepath
0.0 setgray
fill
/Courier findfont
14 scalefont
setfont
60 45 moveto
(Car 1) show

showpage
```

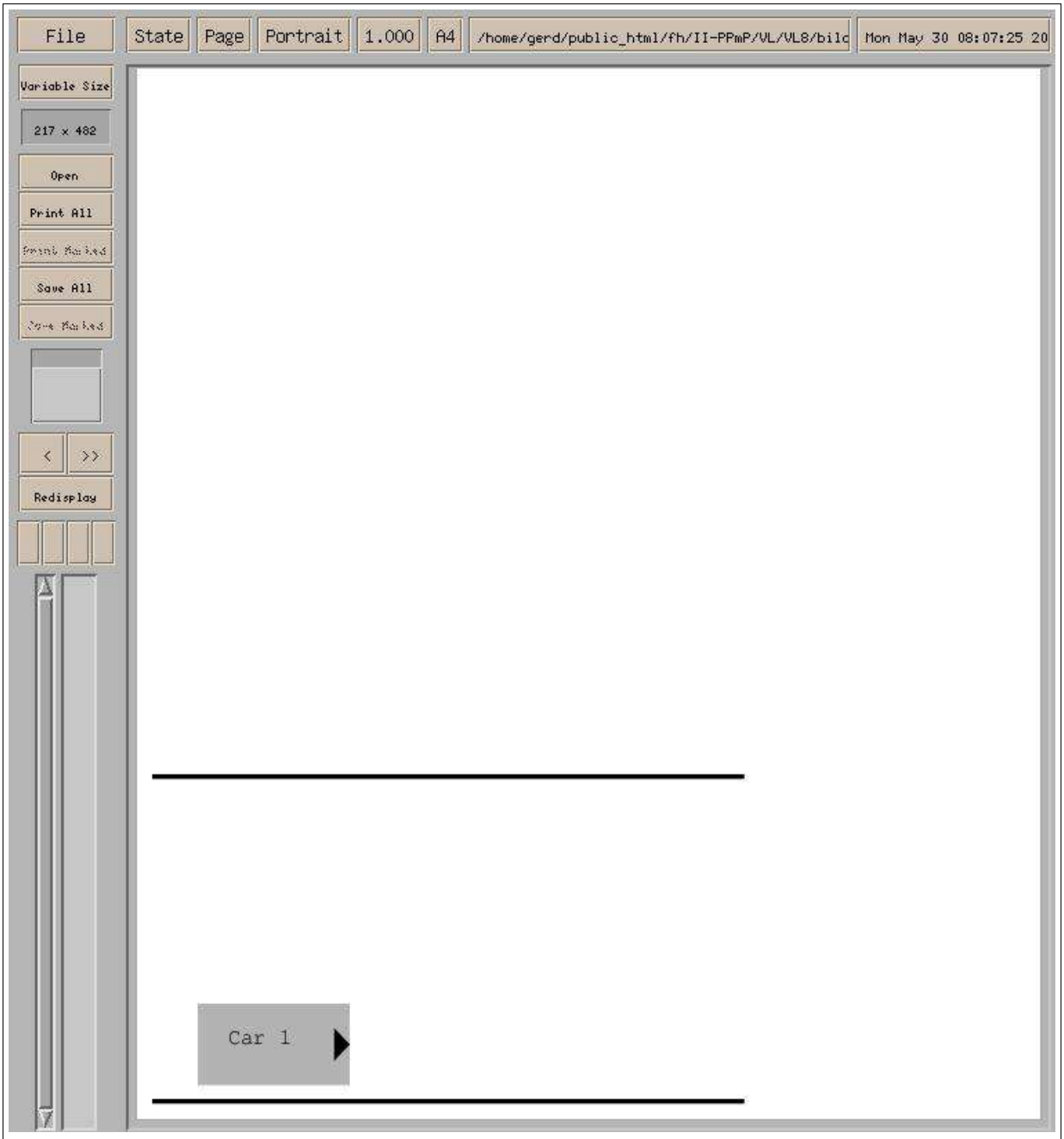

Bild 7: Postscriptprogramm mit "Strasse" und "Auto"

solche Sachverhalte beschreibt, wäre umgangssprachlich der folgende:

Auf der Strasse befindet sich auf der rechten Seite ein Auto mit Bezeichnung Car 1.

In einer etwas normierteren Beschreibung könnte man sich etwa folgendes vorstellen:

Auto Car1 auf Strasse S1 Richtung Osten bei Markierung M5

Auto Car2 auf Strasse S2 Richtung Norden zwischen Markierung M3 und M4