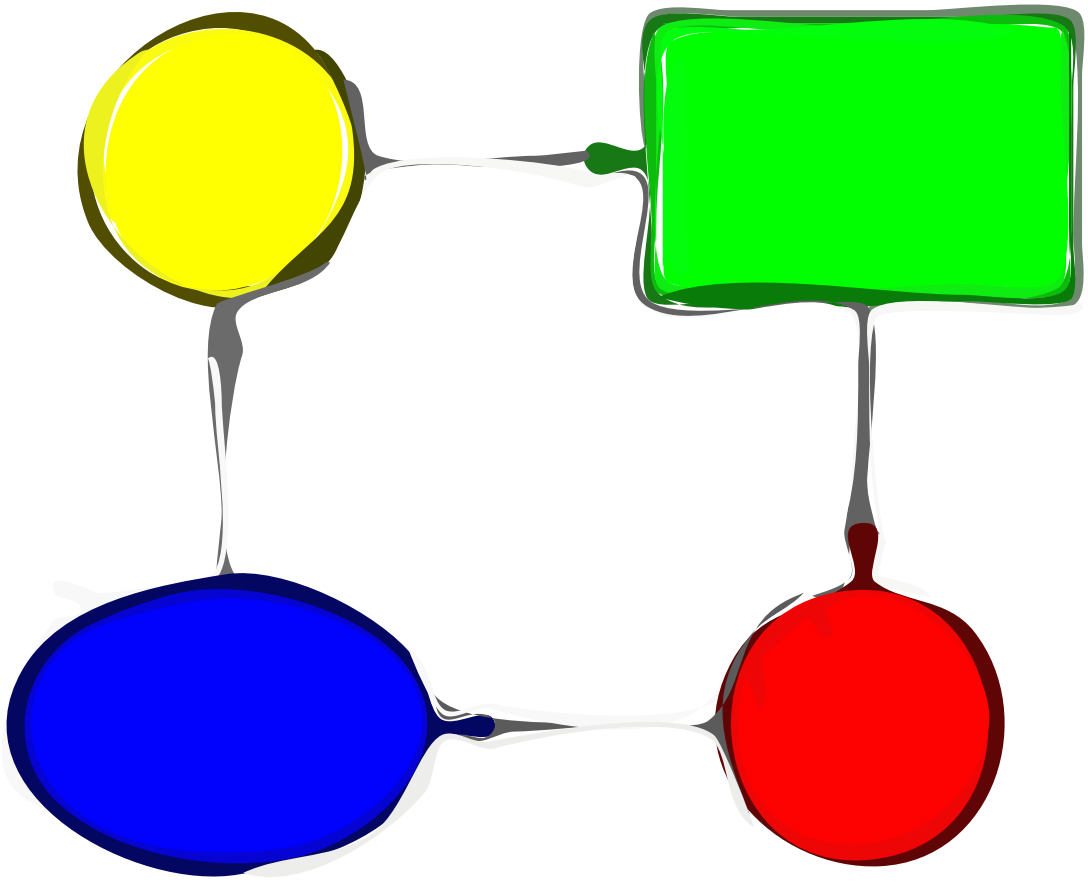
Disegnare con Pic

Articolo estratto dall'opera «Informatica per sopravvivere»

Massimo Piai ⟨*pxam67 (@) virgilio·it*⟩

2006.01.30



Massimo Piai è un matematico appassionato di informatica, che ha trovato nel software libero e nella libertà delle informazioni l'unica possibilità di sviluppare tale passione. I suoi campi di interesse attuali sono la matematica e le scienze, la diffusione della Cultura Informatica, la didattica e la pedagogia.

Il presente lavoro è stato realizzato utilizzando Alml, il sistema di composizione SGML realizzato da Daniele Giacomini per la gestione dei suoi *Appunti di informatica libera*.

Indice generale

Introduzione

La maggior parte delle persone è incapace di disegnare a mano libera anche una semplice linea retta, per non parlare di figure o grafici più complessi; tali persone, pertanto, tendono ad affidarsi a descrizioni verbali delle immagini che vorrebbero disegnare.[1] Con l'ausilio di qualcuno che sia dotato di capacità grafiche è allora possibile ottenere un disegno di buona qualità, a patto di riuscire a comunicare al disegnatore ciò che si intende con un sufficiente grado di chiarezza.

Pic è un preprocessore per Troff (v. sezione 23) mediante il quale è possibile ottenere immagini a partire da una descrizione testuale.

---

[1] C'è poi il caso di coloro che, essendo disabili, hanno un impedimento fisico all'azione stessa del disegnare.

Riconoscimento storico

Il programma Pic originale fu scritto da Brian Kernighan, come complemento al programma Troff di Joseph Ossanna; fu successivamente riscritto da Kernighan, con miglioramenti sostanziali.

Il linguaggio trae ispirazione da linguaggi grafici più antichi, come Ideal e Grap. Kernighan dà credito a Chris van Wyk (l'autore di Ideal) per molte delle idee che sono poi confluite in Pic.

La descrizione originale del linguaggio Pic fu data da Kernighan in una relazione tecnica (vedi sezione 23) di cui esistono due revisioni (anno 1984 e anno 1991).

Esiste una realizzazione GNU di Pic, ossia Gpic, scritta da James Clark ⟨*jjc (ad) jclark·com* ⟩ come parte di Groff;[1] al momento della stesura del presente lavoro, la manutenzione è affidata a Ted Harding ⟨*ted·harding (ad) nessie·mcc·ac·uk* ⟩ e a Werner Lemberg ⟨*wl (ad) gnu·org* ⟩.

[1] **GNU Groff** GNU GPL

Natura di Pic

Pic è un preprocessore che estende le funzionalità di Troff al fine di consentire in modo semplice di tracciare il tipo di diagramma «a scatole e frecce» che si incontra comunemente negli articoli e nei libri di testo di argomento tecnico.

Il linguaggio Pic permette di descrivere in modo procedurale i diagrammi da tracciare, al fine di includere le immagini ottenute nei documenti generati da Troff (oppure da TeX o, in alternativa, per generare file grafici autonomi da utilizzare in seguito, come specificato nella sezione 22). Il linguaggio è abbastanza flessibile da consentire il tracciamento di diagrammi di stato, reti di Petri, [1] diagrammi di flusso, semplici schemi circuitali, mappe concettuali, [2] organigrammi e in generale qualsiasi tipo di illustrazione che richieda l'utilizzo ripetitivo di semplici forme geometriche e curve. La descrizione dei diagrammi è procedurale e orientata agli oggetti, pertanto produce una notazione compatta e semplice da modificare.

Il preprocessore Pic ha un duplice scopo: da un lato permette la descrizione e la generazione di semplici diagrammi e grafici mediante l'uso di un linguaggio naturale; dall'altro consente l'utlizzo di costrutti simili a un normale linguaggio di programmazione, consentendo di ottenere descrizioni molto sintetiche (naturalmente pagando un certo prezzo alla leggibilità; questa possibilità risulta in realtà particolarmente utile se si considera che è possibile preparare l'input per Pic utilizzando un programma esterno).

Il modo migliore per imparare Pic è quello di procedere per tentativi, osservare l'output, eventualmente apporre correzioni e modifiche al sorgente e iterare il procedimento sino al raggiungimento dell'obiettivo prefissato.

Essendo un preprocessore, Pic rielabora alcune specifiche parti di un sorgente Troff prima che quest'ultimo possa procedere ad ulteriori elaborazioni; in pratica Pic traduce queste parti in codici di formattazione di basso livello che Troff può a sua volta comprendere ed elaborare.

Esattamente come nel caso di Tbl e Eqn, una speciale coppia di macro delimita la parte di input che Pic deve elaborare, come segue:

```
.PS
descrizione_pic
.PE
```

Per l'utilizzo effettivo del preprocessore, si veda la sezione 22.

[1] ⟨*http://en.wikipedia.org/wiki/Petri_net* ⟩

[2] ⟨*http://en.wikipedia.org/wiki/Concept_map* ⟩

Versioni di Pic

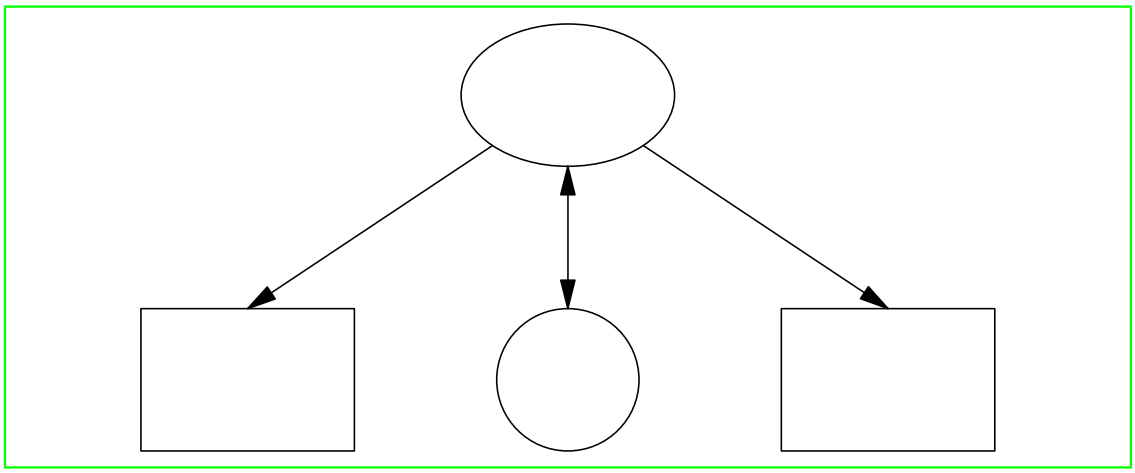
Pic nasce nell'anno 1984, ma la versione originale va oggi ritenuta obsoleta. Pic venne riscritto nell'anno 1991 come parte del pacchetto DBW (*Documenter's Workbench*) per lo Unix System V di AT&T.

Ove fosse necessario evidenziare le differenze fra Pic dell'anno 1991 e la realizzazione GNU (Gpic), verranno nel seguito utilizzati i nomi DWB Pic e GNU Gpic rispettivamente. In ogni caso gli esempi mostrati sono stati sperimentati tramite GNU Gpic.

Un esempio introduttivo

Si immagini di dover descrivere a un interlocutore telefonico la figura 4.1.

Figura 4.1. Primo esempio di figura Pic.



Si potrebbe dire:

> In alto c'è un'ellisse. Ci sono delle frecce collegate a due riquadri e anche a un cerchio...

Mettendosi nei panni di chi deve effettivamente tracciare il disegno dall'altro capo della linea telefonica, ci si rende subito conto delle difficoltà che sorgono se ci si sforza di essere precisi nella descrizione:

> Per prima cosa disegna un'ellisse. Poi scendi sotto l'ellisse e disegna un cerchio. Poi disegna un riquadro a sinistra, e un'altro identico a destra del cerchio. Poi traccia una freccia dal punto in basso a sinistra dell'ellisse al punto in alto del riquadro sinistro. Poi traccia una freccia dal punto in basso a destra dell'ellisse al punto in alto del riquadro destro. Infine traccia una doppia freccia che collega il punto in basso dell'ellisse con il punto in alto del cerchio.

Il listato 4.2 presenta la descrizione equivalente nel linguaggio Pic.

Listato 4.2. Primo esempio di descrizione Pic.

```
.PS
ellipse
move down from bottom of last ellipse
circle
move left from left of last circle
box
move right from right of last circle
box
arrow from lower left of last ellipse to top of 1st box
arrow from lower right of last ellipse to top of 2nd box
arrow <-> from bottom of last ellipse to top of last circle
.PE
```

Anche senza conoscere il linguaggio Pic, se si ha una certa familiarità con la lingua inglese è comunque possibile comprendere la descrizione. Si menzionano alcuni oggetti: un'ellisse (‘**ellipse**’), due riquadri (‘**box**’), un cerchio (‘**circle**’) e tre frecce (‘**arrow**’). Vengono specificati degli spostamenti (‘**move**’) e dei cambiamenti di direzione (‘**down**’, ‘**left**’, ‘**right**’). Inoltre vengono disposti alcuni oggetti relativamente alla posizione di alcuni altri, posizionando i riquadri a sinistra (‘**from left of last circle**’) e a destra (‘**from right of last circle**’) del cerchio e tracciando le frecce fra i vari oggetti (‘**from lower left of last ellipse to top of 1st box**’, ‘**from lower right of last ellipse to top of 2nd box**’, ‘**<-> from bottom of last ellipse to top of last circle**’).

Dall'esempio illustrato dovrebbe emergere il tono generale del linguaggio Pic. In generale si può affermare che la complessità della descrizione va di pari passo con la complessità della figura che si intende ottenere.

Concetti fondamentali

Un sorgente Pic è come un piccolo programma che contiene la ***descrizione*** di una figura. Pic compila tale programma traformandolo in delle macro per Troff. Di conseguenza, qualsiasi altro programma che debba elaborare l'output del programma Troff può tranquillamente ignorare il fatto che originariamente il sorgente conteneva codice Pic.

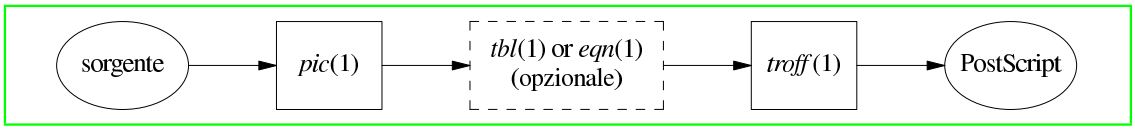
Pic tenta dunque di tradurre tutto ciò che si trova compreso fra le macro '**`.PS`**' e '**`.PE`**', lasciando invariato tutto il resto. In realtà, le suddette macro sono definite nel pacchetto di macro '**`ms`**' (e altrove) e hanno l'effetto di centrare orizzontalmente l'output di Pic.

Le immagini sono descritte in modo procedurale, come collezioni di oggetti collegati da movimenti. Normalmente, Pic cerca di allineare gli oggetti da sinistra a destra nell'ordine di descrizione, collegandoli in punti che siano visivamente naturali. Il listato 5.1 e la figura 5.2 illustrano, ad esempio, il flusso dei dati nell'elaborazione di un sorgente Pic.

Listato 5.1.

```
.PS
ellipse "sorgente";
arrow;
box width 0.6 "\fIpic\/\fP(1)"
arrow;
box width 1.1 "\fItbl\/\fP(1) or \fIeqn\/\fP(1)" "(opzionale)" dashed;
arrow;
box width 0.6 "\fItroff\/\fP(1)";
arrow;
ellipse "PostScript"
.PE
```

Figura 5.2. Il flusso dei dati nell'elaborazione di un sorgente (Pic).

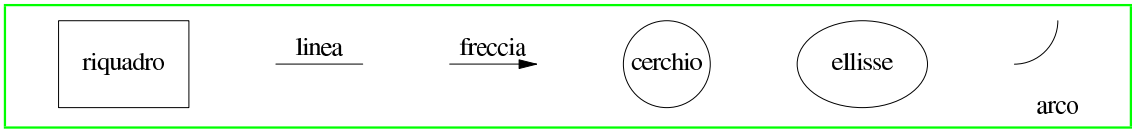


Il listato 5.1 illustra già parecchi concetti fondamentali di Pic: per prima cosa l'invocazione di tre degli oggetti fondamentali (ellisse, freccia, riquadro); poi la dichiarazione di righe di testo da inserire negli oggetti (si noti che si può anche indicare il tipo di carattere); poi l'indicazione di uno stile tratteggiato per un oggetto; infine l'allargamento di un riquadro per adattarlo al testo incluso.

Dal punto di vista sintattico, si noti che le istruzioni terminano con il carattere *newline* oppure con il carattere punto e virgola ('**`;`**'); gli argomenti testuali devono essere inclusi fra apici doppi ('**`"`**'); in generale l'ordine degli argomenti e dei modificatori (come '**`width 1.2`**' o '**`dashed`**') è ininfluente, tranne nel caso degli argomenti testuali.

La figura 5.3 presenta sei dei sette oggetti Pic fondamentali, alla loro dimensione predefinita.

Figura 5.3. Sei dei sette oggetti Pic fondamentali.



L'oggetto mancante è ***spline***; inoltre esiste una maniera per raccogliere vari oggetti in un ***blocco composto*** in modo da trattare per molti versi il gruppo come un oggetto singolo (similmente a un

riquadro). Se ne parlerà nelle sezioni 7.3 e 13.2, rispettivamente.

Gli oggetti ***riquadro***, ***ellisse***, ***cerchio*** e ***blocco composto*** sono ***oggetti chiusi***; gli oggetti ***linea***, ***freccia***, ***arco*** e ***spline*** sono ***oggetti aperti***. Tale distinzione risulta importante per comprendere il funzionamento dei modificatori dei comandi.

La figura 5.3 è stata creata mediante il codice presentato nel listato 5.4, nel quale vengono introdotti alcuni altri concetti fondamentali.

Listato 5.4.

```
1   .PS
2   box "riquadro";
3   move;
4   line "linea" "";
5   move;
6   arrow "freccia" "";
7   move;
8   circle "cerchio";
9   move;
10  ellipse "ellisse";
11  move;
12  arc; down; move; "arco"
13  .PE
```

La prima cosa da notare è il comando '**`move`**', il quale esegue uno spostamento nella direzione corrente di una lunghezza predefinita (0,5 in).

Si noti inoltre come sia possibile arricchire linee e frecce mediante del testo. In questo esempio i comandi '**`line`**' e '**`arrow`**' ricevono entrambi due argomenti, i quali specificano rispettivamente il testo che andrà al di sopra e al di sotto dell'oggetto. Il motivo per cui un argomento testuale solo non va bene dovrebbe essere chiaro osservando l'output di '**`arrow "Ahi!"`**' (figura 5.5).

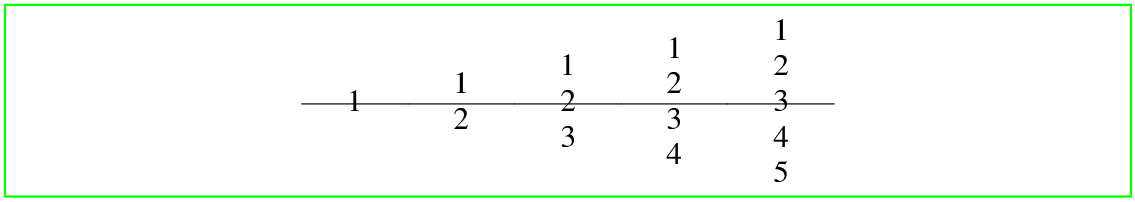
Figura 5.5. Testo centrato su una freccia.

Quando un comando riceve una stringa di testo, Pic tenta di posizionarla al centro geometrico dell'oggetto; aggiungendo più stringhe, si ottiene un blocco verticale che viene complessivamente centrato; si osservi ad esempio il codice del listato 5.6 e la corrispondente figura 5.7.

Listato 5.6.

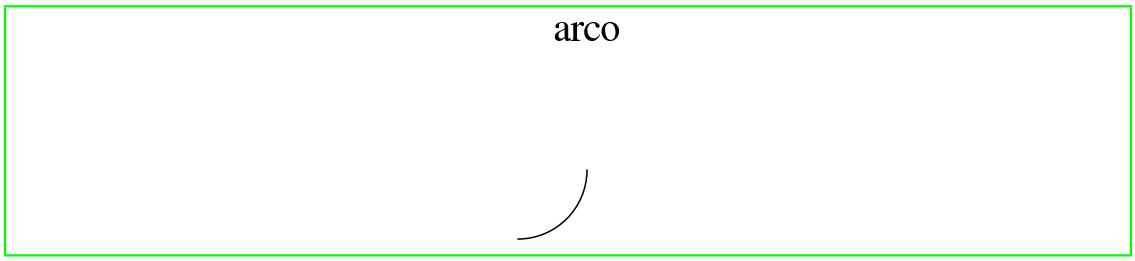
```
.PS
line "1";
line "1" "2";
line "1" "2" "3";
line "1" "2" "3" "4";
line "1" "2" "3" "4" "5";
.PE
```

Figura 5.7. Effetto di argomenti testuali multipli.



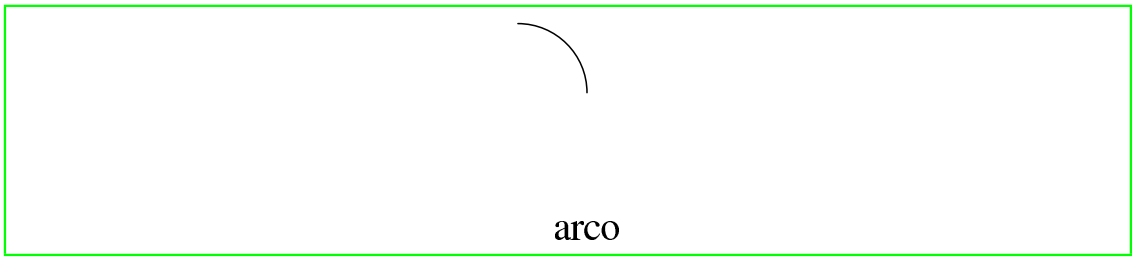
La riga 12 del listato 5.4, la quale descrive l'arco con didascalia, introduce alcuni ulteriori concetti. Si osservi intanto come sia possibile variare la direzione di congiungimento fra i vari oggetti: omettendo l'argomento '**down**', la didascalia si sarebbe congiunta al di sopra dell'arco (figura 5.8).

Figura 5.8. Output di '`arc; move; "arco"`'.



La causa del suddetto comportamento risiede nel fatto che il tracciamento di un arco modifica la direzione corrente in quella verso cui punta il suo estremo finale. Se la cosa non fosse ancora chiara si consideri la figura 5.9.

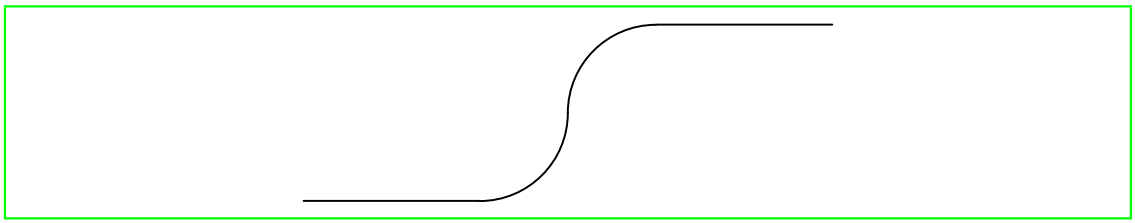
Figura 5.9. Output di '`arc cw; move; "arco"`'.



Nella figura 5.9 l'unica differenza rispetto alla figura 5.8 sta nell'uso dell'argomento '**cw**', che sta per *clockwise* ossia «verso orario» ('**ccw**' invece sta per *counter-clockwise* ossia «verso antiorario»). Si osservi come ciò cambi la direzione corrente verso il basso, piuttosto che verso l'alto.

Un'altra maniera per rendersi conto di quanto sopra osservato è presentata in figura 5.10.

Figura 5.10. Output di '`line; arc; arc cw; line`'.



Si noti che nella figura 5.10 non è stato necessario specificare la direzione verso l'alto per il secondo arco per ottenere una giunzione «liscia» con il primo.

Si osservi infine che una stringa isolata viene considerata come se fosse circondata da un riquadro invisibile dalle dimensioni specificate dagli attributi '**width**' e '**height**' oppure dai valori

predefiniti delle variabili ‘**textwid**’ e ‘**textht**’ (il cui valore iniziale è zero per entrambe, visto che non si può prevedere la dimensione dei caratteri).

Dimensioni e spaziature

Le dimensioni sono implicitamente specificate in pollici (*inch*). Non volendo utilizzare i pollici, è possibile impostare la variabile '**`scale`**' per cambiare l'unità di misura. Impostando '**`scale=2.54`**' si passa in pratica da pollici a centimetri.[1]

6.1 Dimensioni predefinite

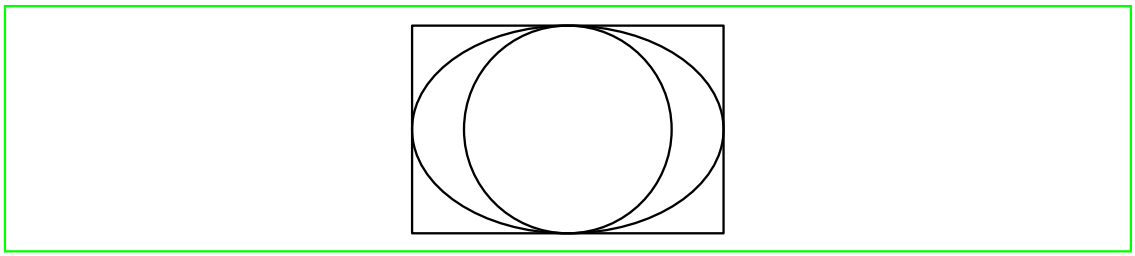
La tabella 6.1 presenta le dimensioni predefinite per gli oggetti Pic.

Tabella 6.1. Dimensioni predefinite per gli oggetti Pic.

Oggetto	Dimensioni predefinite (in pollici)
Riquadro ('**`box`**')	Larghezza: 0,75, altezza: 0,5
Cerchio ('**`circle`**')	Diametro: 0,5
Ellisse ('**`ellipse`**')	Larghezza: 0,75, altezza: 0,5
Arco ('**`arc`**')	Raggio: 0,5
Linea ('**`line`**')	Lunghezza: 0,5
Freccia ('**`arrow`**')	Lunghezza: 0,5

Il modo più intuitivo per considerare i valori predefiniti è osservare che tutti gli oggetti, per difetto, occupano comodamente l'interno di un riquadro standard (figura 6.2).

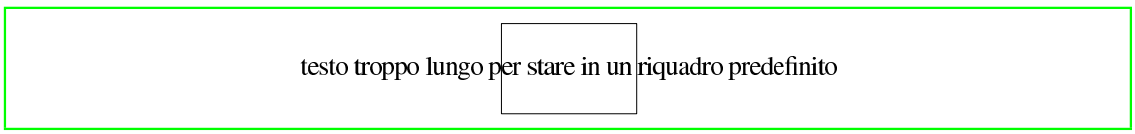
Figura 6.2. Output di '**`box; circle at center of last box; ellipse at center of last box`**'.



6.2 Gli oggetti non sono elastici

Il testo viene presentato nella fonte tipografica corrente con spaziatura predefinita da Troff. Riquadri, cerchi ed ellissi **non** si adattano automaticamente al testo racchiuso. Si osservi l'output di '**`box "testo troppo lungo per stare in un riquadro predefinito"`**', presentato in figura 6.3: probabilmente non è ciò che di solito si intende ottenere.

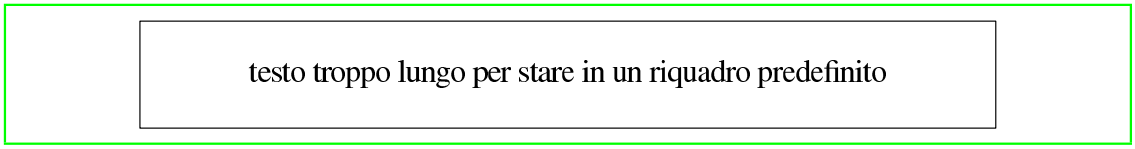
Figura 6.3. I riquadri non si adattano automaticamente.



6.3 Ridimensionamento dei riquadri

Per cambiare le dimensioni di un riquadro è possibile specificarne la larghezza mediante il modificatore '**width**' (figura 6.4).

Figura 6.4. Output di '**box width 4 "testo troppo lungo per stare in un riquadro predefinito"**'.

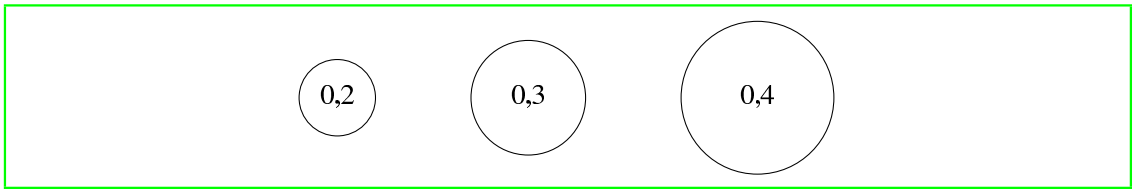


Il modificatore accetta una dimensione espressa in pollici. È previsto anche il modificatore '**height**' il quale modifica l'altezza del riquadro. Le parole chiave '**width**' e '**height**' possono essere abbreviate mediante '**wid**' e '**ht**' rispettivamente.

6.4 Ridimensionamento di altri oggetti

Per modificare le dimensioni di un cerchio si utilizzano i modificatori '**radius**' o '**diameter**' (abbreviabili in '**rad**' o '**diam**', rispettivamente); a seconda del valore numerico successivamente indicato si ottiene una modifica del raggio o del diametro del cerchio, rispettivamente (figura 6.5).

Figura 6.5. Cerchi di raggio crescente.

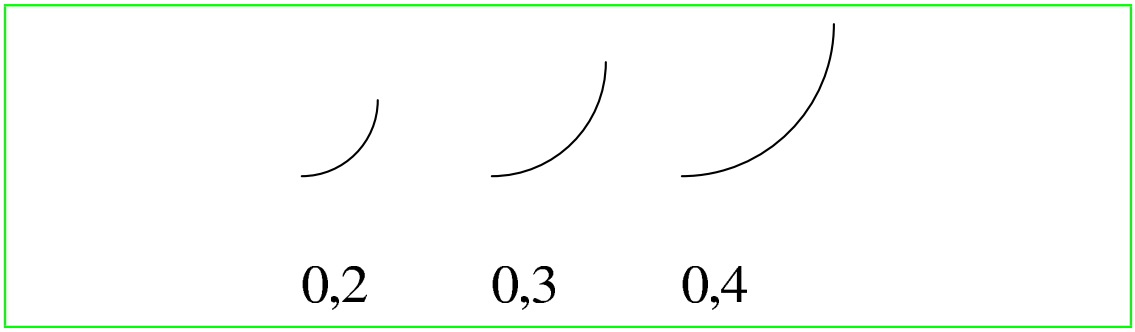


Anche il comando '**move**' accetta una dimensione, la quale indica di quanti pollici ci si deve spostare nella direzione corrente.

Le ellissi hanno dimensioni tali da essere inscritte nel riquadro rettangolare definito dai loro assi, e possono essere ridimensionate tramite '**width**' e '**height**', come i riquadri.

È possibile modificare il raggio di curvatura di un arco tramite '**radius**' (o '**rad**'), il quale specifica il raggio del cerchio cui l'arco appartiene. Al crescere del valore indicato si ottengono archi più piatti (figura 6.6).

Figura 6.6. Archi di raggio crescente.

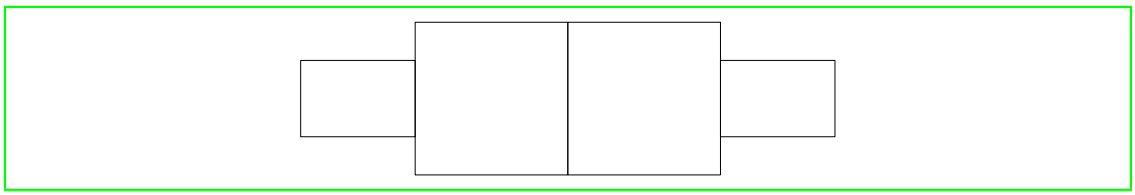


Si noti che, a causa del fatto che un arco equivale a un quarto di cerchio, aumentando il raggio si aumenta anche la dimensione del riquadro circoscritto all'arco.

6.5 La parola chiave «same»

Al posto dell'indicazione esplicita delle dimensioni è possibile utilizzare la parola chiave '**`same`**'. In tal modo l'oggetto avrà le medesime dimensioni dell'oggetto (dello stesso tipo) che precede (figura 6.7).

Figura 6.7. Output di '**`box; box wid 1 ht 1; box same; box`**'.



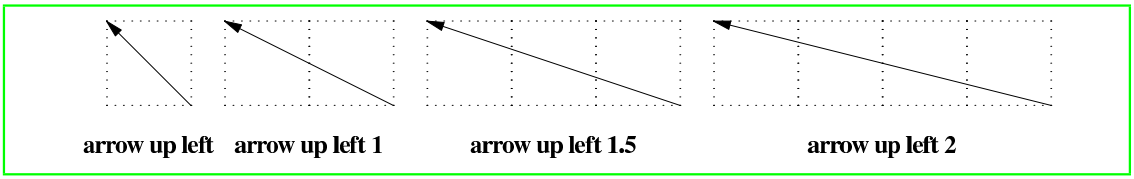
[1] Si tenga presente che il cambiamento di scala ha effetto solamente sui valori predefiniti delle variabili di stile; v. anche sezione 14.

Linee generalizzate e spline

7.1 Linee diagonali

È possibile specificare linee o frecce diagonali aggiungendo uno o più modificatori '**`up`**', '**`down`**', '**`left`**' oppure '**`right`**' all'oggetto. Ognuno dei suddetti modificatori può presentare un coefficiente moltiplicatore. Per comprenderne l'effetto si immagini la superficie grafica con una griglia di riquadri standard (figura 7.1).

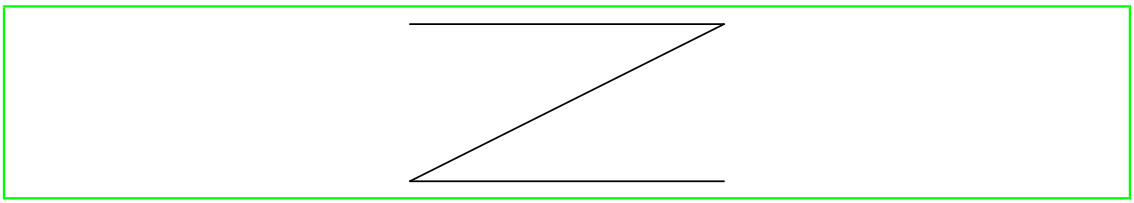
Figura 7.1. Frecce diagonali (i riquadri punteggiati mostrano la griglia sottintesa, spaziata di mezzo pollice).



7.2 Spezzate

Un oggetto linea o freccia può anche consistere di una spezzata composta da segmenti di diversa lunghezza e direzione. Per descrivere una spezzata si possono collegare diversi comandi '**`line`**' o '**`arrow`**' mediante la parola chiave '**`then`**' (figura 7.2).

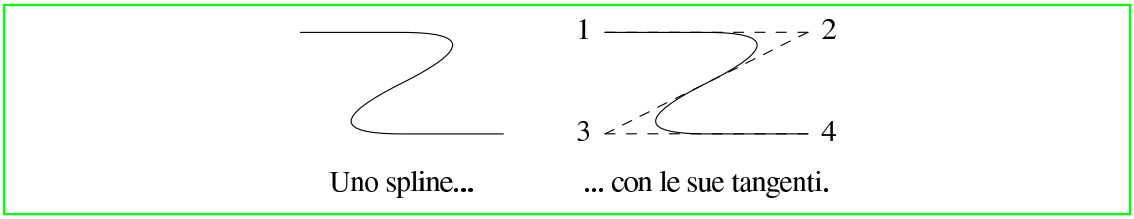
Figura 7.2. Output di '`line right 1 then down .5 left 1 then right 1`'.



7.3 Spline

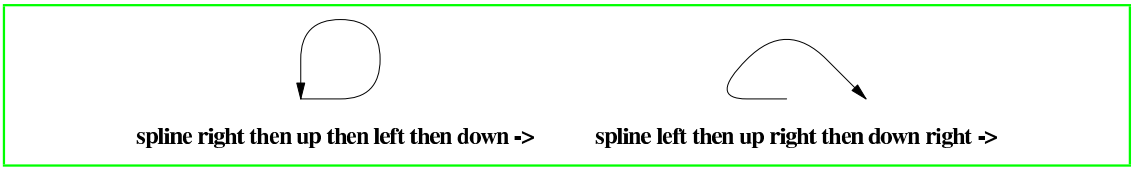
Cominciando una spezzata con la parola chiave '**`spline`**' si ottiene che i vertici della spezzata vengano considerati punti di controllo per una curva regolare interpolatrice (figura 7.3).

Figura 7.3. '`spline right 1 then down .5 left 1 then right 1`'.



Mediante gli spline è possibile descrivere molte curve irregolari ma dall'aspetto naturale (figura 7.4).

Figura 7.4. Altri due spline di esempio.



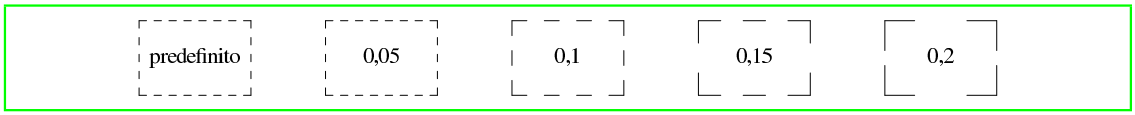
Si noti nella figura 7.4 la decorazione a freccia (‘**->**’). La punta della freccia si può applicare in modo naturale agli oggetti a spezzata, sia basati su linee che su spline (si veda in proposito la sezione 8).

Decorazioni

8.1 Oggetti tratteggiati

È stato già osservato che il modificatore '**dashed**' può cambiare lo stile delle linee di un oggetto da solido a tratteggiato. GNU Gpic permette di punteggiare o tratteggiare ellissi,[1] cerchi e archi (e anche gli spline in modalità TeX, vedi sezione 20); alcune versioni di DWB Pic permettono solamente il tratteggio di linee e riquadri. È anche possibilile variare l'intervallo del tratteggio specificando un valore numerico subito dopo il modificatore (figura 8.1).

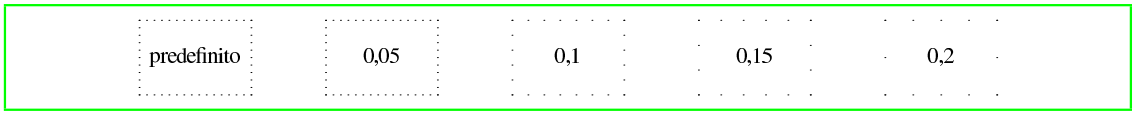
Figura 8.1. Oggetti tratteggiati.



8.2 Oggetti punteggiati

Un ulteriore modificatore disponibile è '**dotted**'. Anche in questo caso è possibile specificare l'intervallo di punteggiatura mediante un valore numerico subito dopo il modificatore (figura 8.2).

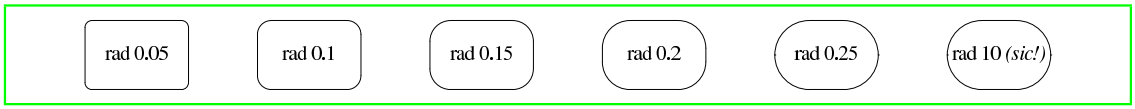
Figura 8.2. Oggetti punteggiati.



8.3 Vertici arrotondati

GNU Gpic permette anche l'arrotondamento degli vertici di un riquadro, mediante il modificatore '**rad**' (figura 8.3).

Figura 8.3. '**box rad**' con valori crescenti del valore numerico.

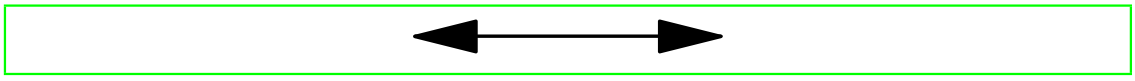


Valori del raggio di curvatura maggiori della metà della più piccola dimensione del riquadro vengono implicitamente troncati a tale valore.

8.4 Punte di freccia

Linee ed archi possono essere a loro volta decorati. Qualsiasi linea o arco (o spline) può essere decorata mediante punte di freccia utilizzando i modificatori '**->**' e '**<-**' (figura 8.4).

Figura 8.4. Doppia freccia realizzata mediante '**line <- ->**'.



Effettivamente il comando '**arrow**' altro non è che un sinonimo di '**line ->**'. Esiste inoltre il modificatore a doppia punta di freccia '**<->**', pertanto la figura 8.4 si può generare col codice '**line <->**'.

Le punte di freccia hanno un attributo '**width**', che specifica la dimensione trasversale della punta, e un attributo '**height**' che ne specifica la dimensione longitudinale.

Lo stile delle punte di freccia è controllato dalla variabile di stile (vedi sezione 14) denominata '**arrowhead**'. Le versioni DWB Pic e GNU Gpic interpretano tale variabile in maniera differente. Per difetto, DWB Pic usa punte di freccia aperte, con valore 2 per la variabile '**arrowhead**'; l'articolo di Kernighan (v. 23) specifica che un valore pari a 7 corrisponde a punte di freccia solide. Viceversa, per difetto, GNU Gpic usa punte di freccia solide e il valore 1 per '**arrowhead**'; il valore 0 corrisponde a punte di freccia aperte.[2]

8.5 Spessori

È anche possibile modificare lo spessore delle linee che compongono un oggetto (si tratta di un'estensione GNU Gpic; DWB Pic **non** prevede tale possibilità). Lo spessore predefinito è controllato dalla variabile '**linethick**'. Lo spessore è espresso in punti. Valori negativi implicano spessore predefinito: in modalità TeX (v.sezione 20), usando l'opzione '**-t**', ciò significa utilizzare uno spessore pari a otto millesimi di pollice; in modalità TeX, usando l'opzione '**-c**', significa lo spessore specificato dal comando '**.ps**' (*Point size*); in modalità Troff significa utilizzare uno spessore proporzionale alle dimensioni dei punti. Valore zero significa tracciare la linea con lo spessore minimo compatibile con il dispositivo di output. Il valore iniziale di '**linethick**' è -1. Esiste anche l'attributo '**thickness**' (abbreviabile in '**thick**'). Per esempio '**circle thickness 1.5**' traccia un cerchio con uno spessore di un punto e mezzo. Lo spessore delle linee non è influenzato dal valore della variabile '**scale**' (sezione 6), né dagli argomenti specificati assieme alla macro '**.PS**' (sezione 19).

8.6 Oggetti invisibili

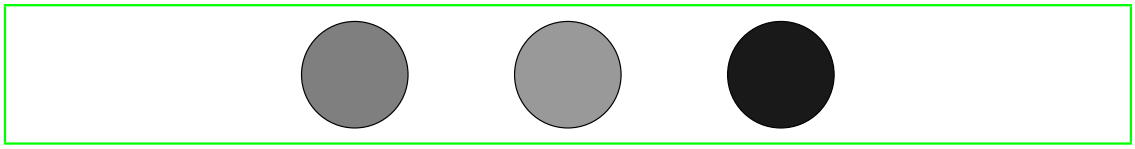
Il modificatore '**invisible**' (abbreviabile in '**invis**') rende un oggetto completamente invisibile. Un uso tradizionale di questa possibilità era quello di piazzare del testo in un oggetto invisibile, per ottenere un posizionamento naturale rispetto agli oggetti circostanti. Le versioni più recenti di DWB Pic e la realizzazione GNU Gpic gestiscono il testo isolato esattamente in questo modo.

8.7 Oggetti campiti

È possibile campire riquadri, cerchi ed ellissi mediante il modificatore '**filled**' (abbreviabile in '**fill**'). Si può anche specificare un valore numerico; il valore predefinito è dato dalla variabile di stile (vedi sezione 14) denominata '**fillval**'.

DWB Pic e GNU Gpic seguono convenzioni opposte per i valori di campitura, e prevedono valori predefiniti differenti. Per DWB Pic il valore predefinito di '**fillval**' è 0,3 e i valori più piccoli corrispondono a campiture più scure; GNU Gpic utilizza zero per il bianco e uno per il nero, e il valore predefinito di '**fillval**' è 0,5 (figura 8.5).

Figura 8.5. '`circle fill; move; circle fill 0.4; move; circle fill 0.9;`'.



GNU Gpic fornisce ulteriori garanzie. È possibile usare un valore di campitura superiore a uno: ciò significa campire con la sfumatura di grigio attualmente usata per il testo e le linee. Ciò

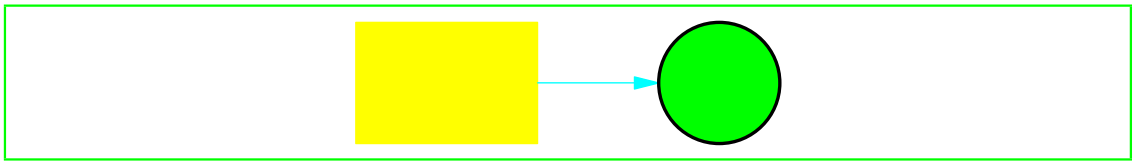
corrisponde normalmente al nero, ma certi dispositivi di output permettono di modificare tale comportamento. L'attributo di invisibilità non influisce la campitura degli oggetti. Qualsiasi testo associato con un oggetto campito viene disegnato dopo che l'oggetto è stato campito, sicché il testo non viene coperto dalla campitura.

Gli oggetti chiusi prevedono il modificatore '**solid**', che equivale a campire con il valore di campitura più scuro

8.8 Oggetti colorati

Come estensione GNU sono previsti altri tre modificatori per specificare il colore degli oggetti. '**outline**' imposta il colore del contorno, '**shaded**' il colore di campitura, '**color**' entrambi i colori. Tutti e tre prevedono un suffisso che specifichi il colore (figura 8.6).

Figura 8.6. `box color "yellow"; arrow color "cyan"; circle shaded "green" thickness 1 outline "black";'.



È prevista anche una sintassi alternativa: '**colour**', '**colored**', '**coloured**', '**outlined**'.

Al momento di questa stesura i colori non sono contemplati in modalità TeX (sezione 20). I nomi standard dei colori per Groff si trovano nei file di macro per i vari dispositivi di output (per esempio '`/usr/share/groff/current/tmac/ps.tmac`' per il dispositivo PostScript).

Pic presuppone che all'inizio di una figura sia il colore di tracciamento che quello di campitura siano impostati al valore predefinito.

[1] Purtroppo, la versione di GNU Gpic provata al momento della presente stesura (parte del pacchetto Groff nella versione 1.18.1-15) non sembra consentire la decorazione delle ellissi.

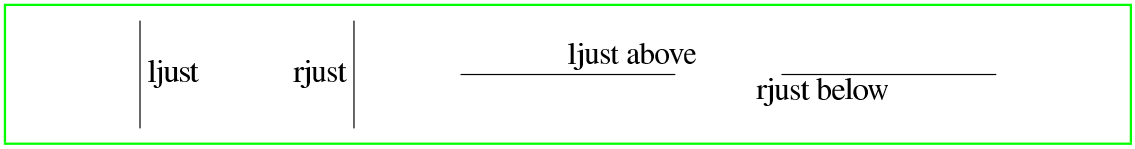
[2] Le punte di freccia solide vengono sempre campite con il colore di contorno corrente (v. sezione 8.8).

Ulteriori indicazioni sul posizionamento del testo

Per difetto il testo viene centrato in corrispondenza del centro geometrico dell'oggetto a cui il testo è associato. Il modificatore '**`ljust`**' fa sì che l'estremo sinistro del testo si trovi nel punto specificato (il che significa che il testo giacerà **alla destra** del punto stesso), mentre il modificatore '**`rjust`**' fa sì che l'estremo destro del testo si trovi nel punto specificato. I modificatori '**`above`**' e '**`below`**' centrano il testo al di sopra e al di sotto dell'oggetto, rispettivamente, a una distanza pari a mezza linea di testo.

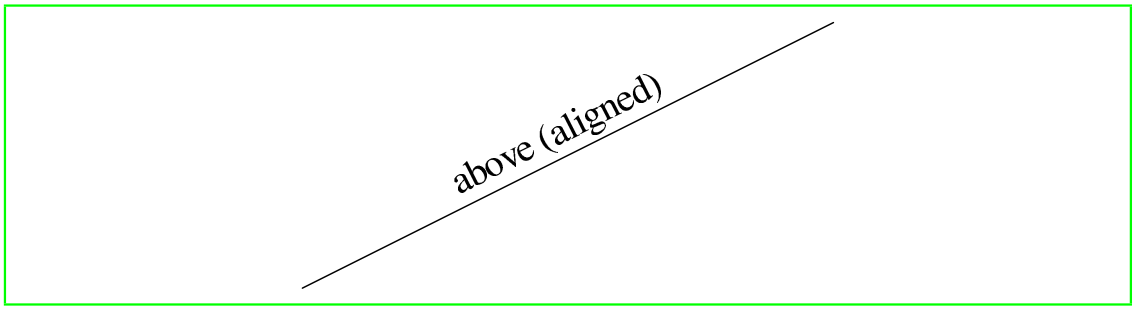
Gli attributi suddetti possono anche essere combinati (figura 9.1).

Figura 9.1. Attributi del testo.



In GNU Gpic gli oggetti possono avere un attributo '**`aligned`**', che ha effetto solo quando il postprocessore è '**`grops`**'.[1] Il testo associato a un oggetto con attributo '**`aligned`**' viene ruotato (figura 9.2) attorno al centro dell'oggetto in modo da risultare allineato nella direzione che va dal punto iniziale al punto finale dell'oggetto stesso.[2]

Figura 9.2. '**`line aligned up 1 right 2 "above (aligned)" above`**'.



[1] situazione predefinita utilizzando Groff

[2] Ovviamente l'attributo non ha effetto per gli oggetti in cui i due punti coincidono.

Sezione 10

Ulteriori indicazioni sui cambiamenti di direzione

Si è già spiegato come cambiare la direzione, lungo la quale gli oggetti vengono composti, da verso destra a verso il basso. Le figure 10.1 e 10.2 illustrano ulteriormente la cosa.

Figura 10.1. Effetti delle diverse direzioni di spostamento (verso destra e verso sinistra).

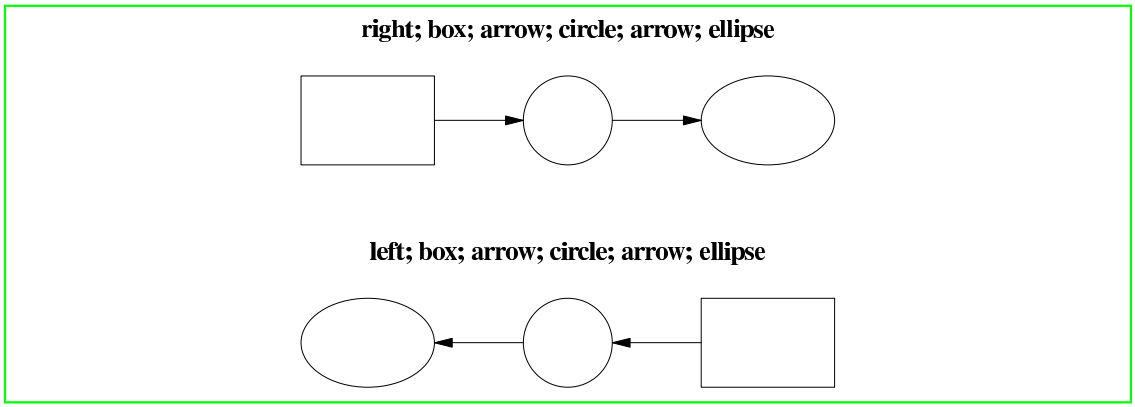
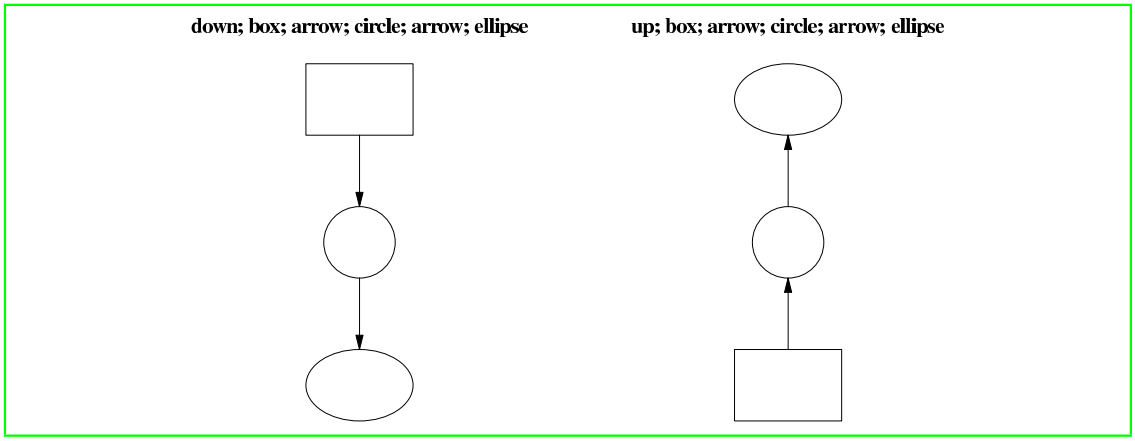
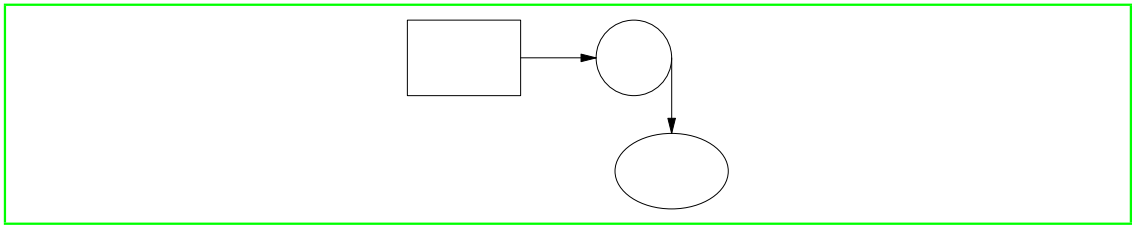


Figura 10.2. Effetti delle diverse direzioni di spostamento (verso il basso e verso l'alto).



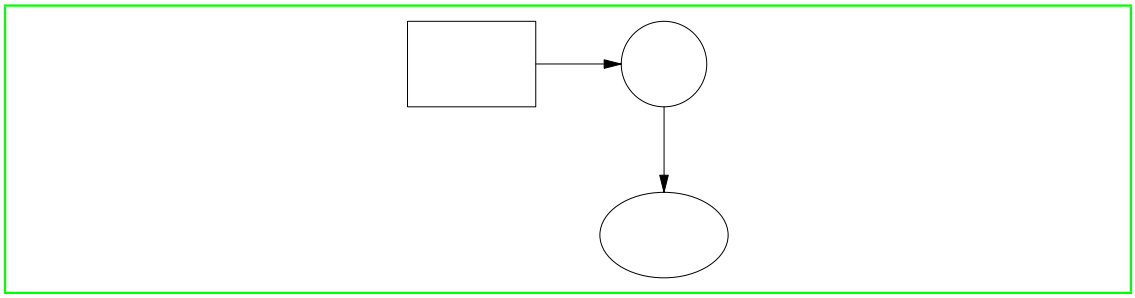
Può accadere qualcosa di sorprendente se si cambia direzione in modo apparentemente ovvio (figura 10.3).

Figura 10.3. '`box; arrow; circle; down; arrow; ellipse`'.



Magari ci si aspettava che il codice '**`box; arrow; circle; down; arrow; ellipse`**' producesse la figura 10.4, ma effettivamente per ottenere tale risultato si deve utilizzare il codice presentato nel listato 10.5. Per quale motivo? Perché il punto di uscita per la direzione corrente è già impostato quando si disegna il nuovo oggetto. La seconda freccia nella figura 10.3 cala verso il basso a partire dal punto del cerchio a cui si dovrebbe congiungere un nuovo oggetto verso destra.

Figura 10.4. Più intuitivo?



Listato 10.5.

```
1  .PS
2  box;
3  arrow;
4  circle;
5  move to bottom of last circle;
6  down;
7  arrow;
8  ellipse
9  .PE
```

Il significato di '**`move to bottom of last circle;`**' dovrebbe essere evidente. Per comprendere la cosa in generale è necessario approfondire due temi importanti: posizioni e nomi degli oggetti (sezioni 12 e 11 rispettivamente).

Nomi

La maniera più naturale per indicare le posizioni in Pic è in relativamente agli oggetti. Per far ciò è necessario poter assegnare dei nomi agli oggetti stessi. Il linguaggio Pic prevede numerose possibilità in tal senso, mutuate in modo naturale della lingua inglese.

11.1 Nominare gli oggetti per ordine di tracciamento

Il modo più semplice (e in genere il più utile) per nominare un oggetto è tramite la clausola '**last**'. È necessario che essa sia seguita dall'indicazione del tipo di oggetto: '**box**', '**circle**', '**ellipse**', '**line**', '**arrow**', '**spline**', '**""**' oppure '**[]**' (l'ultimo tipo si riferisce a un ***blocco composto***, di cui si discuterà nella sezione 13.2). Pertanto, ad esempio, la clausola '**last circle**' nel listato 10.5 si riferisce all'ultimo cerchio tracciato.

Generalizzando, si può dire che gli oggetti di un dato tipo sono implicitamente numerati a partire da 1. Ci si può riferire per esempio alla terza ellisse nella figura corrente tramite '**3rd ellipse**', oppure al primo riquadro tramite '**1st box**', o ancora alla quinta stringa di testo (la quale non sia attributo di un altro oggetto, naturalmente) tramite '**5th ""**'.

Inoltre gli oggetti sono numerati alla rovescia, per ciascun tipo, a partire dall'ultimo. Si può quindi utilizzare '**2nd last box**' per indicare il penultimo riquadro, oppure '**3rd last ellipse**' per la terz'ultima ellisse.

Ove sia prevista la notazione seguente:

```
nth
```

si può usare anche la seguente:

```
'espressione'th
```

ove ***espressione*** abbia valore numerico (intero).[1] Per un esempio, si consideri il listato 11.1.

Listato 11.1.

```
...
for i = 1 to 4 do {
   line from upper left of 'i'th box to lower right of 'i+1'th box
}
...
```

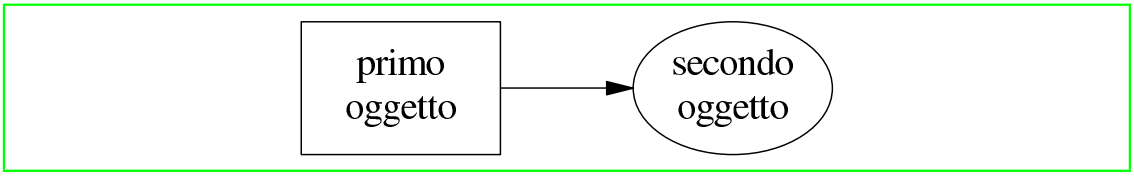
11.2 Nominare gli oggetti mediante etichette

Si possono menzionare gli oggetti anche tramite delle etichette. Un'***etichetta*** è una parola (che inizi con una lettera maiuscola) seguita dai due punti ('**:**'); viene dichiarata semplicemente posizionandola immediatamente prima del comando di tracciamento dell'oggetto. Ad esempio, il codice del listato 11.2 dichiara le etichette '**A**' e '**B**' per il primo e il secondo oggetto, rispettivamente; la figura 11.3 presenta il risultato.

Listato 11.2.

```
1  .PS
2  A: box "primo" "oggetto"
3  move;
4  B: ellipse "secondo" "oggetto"
5  move;
6  arrow right at right of A;
7  .PE
```

Figura 11.3. Esempio di utilizzo delle etichette.



L'istruzione '**at**' nella riga 6 utilizza l'etichetta '**A**' (il comportamento di '**at**' verrà spiegato nella sezione 12). In seguito (sezione 13.2) si evidenzierà quanto le etichette si rivelino particolarmente utili nel caso degli oggetti composti.

Le etichette non sono costanti, bensì variabili.[2] Con del codice come '**A: A + (1,0);**' si ottiene l'effetto di riassegnare all'etichetta '**A**' un valore che designa una nuova posizione che si trova un pollice a destra della vecchia (v. anche sezione 12).

[1] '**'th**' è un singolo elemento sintattico: non ci deve essere uno spazio fra '**'**' e '**th**'.

[2] Si può considerare l'elemento sintattico '**:**' come una spece di operatore di assegnamento.

Posizioni

La posizione dei punti sulla superficie grafica può venire descritta in molti modi differenti. Per quanto riguarda la sintassi del linguaggio Pic, le varie forme descrittive sono equivalenti: ove se ne possa usare una in particolare, ogni altra che abbia la stessa semantica è permessa anch'essa.

L'etichetta speciale '**`Here`**' si riferisce sempre alla posizione corrente.

12.1 Coordinate assolute

Il modo più immediato per indicare un punto è utilizzare le coordinate cartesiane assolute;[1] Pic utilizza un sistema di riferimento cartesiano con origine nel vertice inferiore sinistro della superficie grafica virtuale (con gli assi orientati in maniera usuale). Una posizione assoluta si può sempre scrivere nel modo usuale come coppia di numeri separati dalla virgola e racchiusi in parentesi (tonde) - e tale pratica è raccomandata per garantire la leggibilità del codice. Se il contesto è tale da escludere ambiguità, la coppia di coordinate può essere indicata senza le parentesi.

L'utilizzo delle coordinate assolute **è tuttavia sconsigliabile**, poiché esse tendono a rendere le descrizioni difficilmente comprensibili e adattabili. È meglio utilizzare la flessibilità del linguaggio Pic per specificare le posizioni **relativamente agli oggetti** precedentemente tracciati (sezione 12.2).

12.2 Posizioni relative agli oggetti

Il simbolo '**`Here`**' si riferisce sempre alla posizione dell'ultimo oggetto tracciato, oppure al punto finale dell'ultimo spostamento.

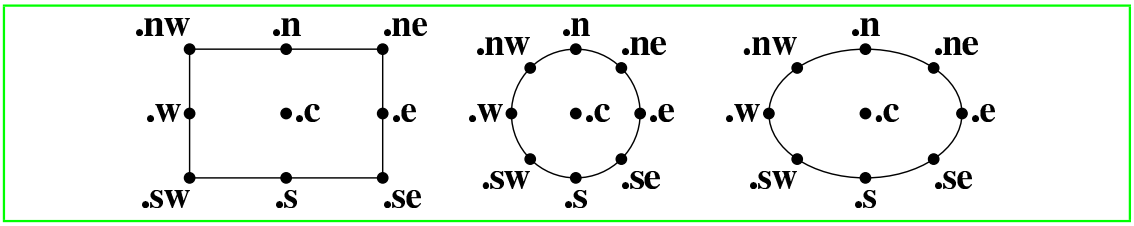
Senza ulteriori specificazioni, una descrizione come '**`last circle`**' come di qualunque oggetto chiuso o arco, si riferisce al centro geometrico dell'oggetto stesso. Senza ulteriori specificazioni, il nome di una linea o spline si riferisce al punto iniziale dell'oggetto stesso.

Inoltre gli oggetti Pic posseggono un certo numero di posizioni menzionabili associate ad essi. Una di queste è il centro dell'oggetto, il quale (per ridondanza) può essere indicato con il suffisso '**`.center`**' (o semplicemente '**`.c`**'). Sicché '**`last circle .center`**' equivale a '**`last circle`**'.

12.2.1 Posizioni relative agli oggetti chiusi

Ciascun oggetto chiuso (riquadri, cerchi, ellissi o blocchi composti) possiede altresì ben otto punti (cardinali) associato ad esso (figura 12.1).

Figura 12.1. I punti cardinali.



Pertanto, il codice '**`last circle .s`**' si riferisce al punto sud dell'ultimo cerchio tracciato. La riga 5 del listato 10.5 si potrebbe dunque scrivere così: '**`move to last circle .s;`**'.

In alternativa ai quattro punti cardinali principali (‘**.n**’, ‘**.s**’, ‘**.e**’ e ‘**.w**’) sono disponibili rispettivamente i nomi ‘**.top**’, ‘**.bottom**’, ‘**.left**’ e ‘**.right**’ (o semplicemente ‘**.t**’, ‘**.b**’, ‘**.l**’ e ‘**.r**’).

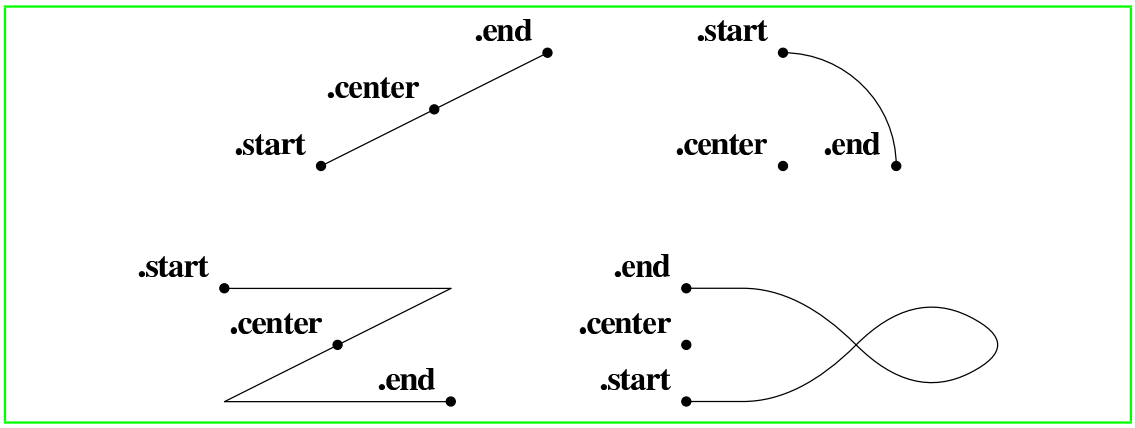
Si possono anche usare i nomi ‘**center**’, ‘**top**’, ‘**bottom**’, ‘**left**’, ‘**right**’, ‘**north**’, ‘**south**’, ‘**east**’ e ‘**west**’ (si noti che manca il punto iniziale) in forma di prefisso marcato dalla parola chiave ‘**of**’; sicché ‘**center of last circle**’ oppure ‘**top of 2nd last ellipse**’ sono entrambi corretti riferimenti a oggetti. Infine, i nomi ‘**left**’ e ‘**right**’ possono essere preceduti da ‘**upper**’ oppure ‘**lower**’, con ovvio significato (vedi anche listato 4.2).

Anche gli archi possiedono punti cardinali, che concidono con quelli dell’associato cerchio.

12.2.2 Posizioni relative agli oggetti aperti

Gli oggetti aperti (linee, frecce, archi e spline) possiedono tre punti menzionabili: ‘**.start**’, ‘**.center**’ (oppure ‘**.c**’) e ‘**.end**’. È possibile anche utilizzarli in forma prefissa (senza il punto iniziale, e con la parola chiave ‘**of**’). Il centro dell’arco coincide con il centro del cerchio associato, ma il centro di una linea, spezzata o spline è **il punto medio degli estremi** (figura 12.2).

Figura 12.2. Punti speciali associati agli oggetti aperti.



12.3 Comporre le posizioni

A partire da due posizioni date, esistono diverse maniere per combinarle insieme per contruirne una terza.

12.3.1 Somma vettoriale e traslazioni

Le posizioni possono essere addizionate o sottratte per generare una nuova posizione.[2] Il risultato è l’usuale somma vettoriale (somma per coordinate). Ad esempio, ‘**last box .ne + (0.1, 0)**’ è una posizione valida: questo esempio costituisce un utilizzo tipico, per definire una posizione lievemente sfasata rispetto a quella data.[3]

12.3.2 Interpolazione

Una posizione può risultare dall'***interpolazione*** di due posizioni date. La sintassi è:

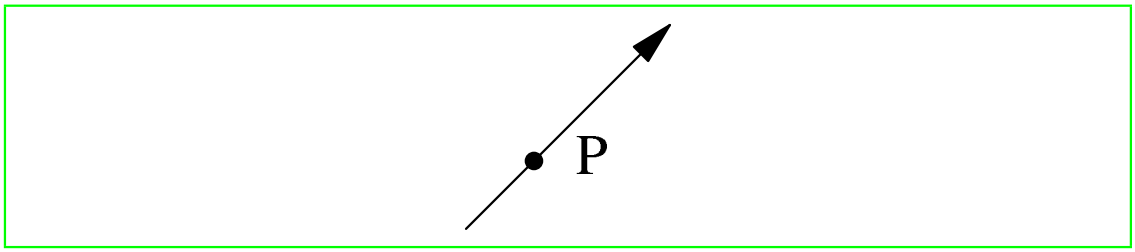
```
frazione of the way between prima_posizione and seconda_posizione
```

Per esempio, si può scrivere: '**`1/3 of the way between here and last ellipse .ne`**'. La frazione può esprimersi nella forma ***numeratore*/*denominatore*** oppure in notazione decimale.[4] Alternativamente, e in modo più sintetico, si può scrivere:

```
frazione <prima_posizione , seconda_posizione>
```

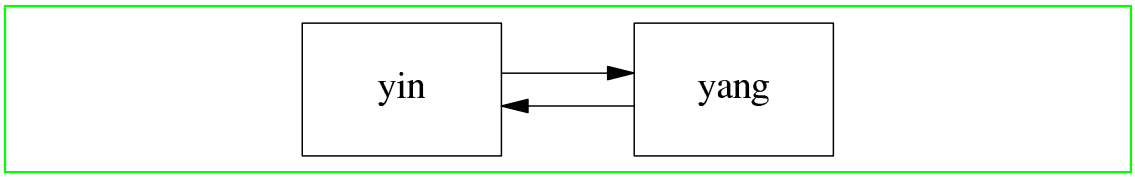
Pertanto il precedente esempio potrebbe scriversi: '**`1/3 <here, last ellipse .ne>`**'. Come ulteriore esempio si consideri la figura 12.3.

Figura 12.3. '**`P: 1/3 of the way between last arrow .start and last arrow .end`**'.



L'interpolazione può essere utile, ad esempio, per tracciare connessioni bidirezionali (figura 12.4).

Figura 12.4. Connessioni bidirezionali.



La figura 12.4 corrisponde al codice del listato 12.5. Si noti l'uso della forma abbreviata per l'interpolazione.

Listato 12.5.

```
A: box "yin"; move;
B: box "yang";
arrow right at 1/4 <A.e,A.ne>;
arrow left  at 1/4 <B.w,B.sw>;
```

12.3.3 Proiezioni

Date due posizioni ***prima_posizione*** e ***seconda_posizione***, la posizione:

```
( prima_posizione , seconda_posizione )
```

ha l'ascissa di ***prima_posizione*** e l'ordinata di ***seconda_posizione***. Ciò può tornare utile per posizionare oggetti ai vertici di un riquadro ipotetico determinato da altri due oggetti (figura 12.6).

Figura 12.6. Utilizzo della composizione per proiezione.

12.4 Utilizzo delle posizioni

Ci sono quattro modi di utilizzo delle posizioni: '**`at`**', '**`from`**', '**`to`**' e '**`with`**'. Si tratta di quattro modificatori, cioè vanno usati come suffisso dei comandi di tracciamento.

Il modificatore '**`at`**' comporta il tracciamento di un oggetto chiuso oppure di un arco con il centro coincidente con la posizione che segue, ossia il tracciamento di una linea, spline o freccia a partire dalla posizione che segue.

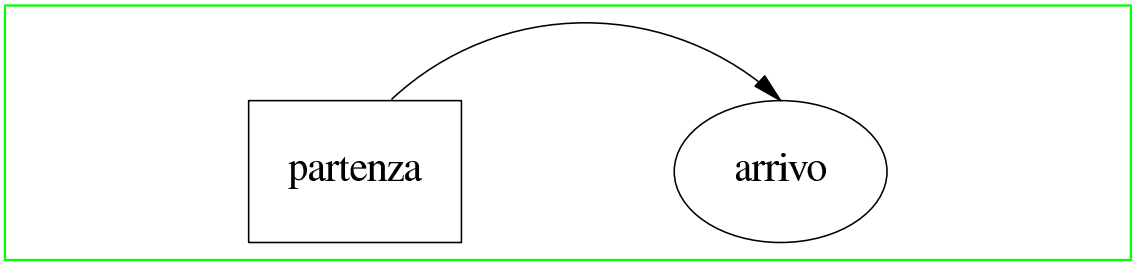
Il modificatore '**`to`**' può essere utilizzato da solo per specificare la destinazione di uno spostamento; analogamente per il modificatore '**`from`**'.

I modificatori '**`from`**' e '**`to`**' possono anche essere utilizzato assieme a '**`line`**' o '**`arc`**' per specificare gli estremi dell'oggetto. Usati assieme ai nomi di posizione, forniscono un meccanismo assai flessibile al fine di connettere i vari oggetti. Si considerino ad esempio il listato 12.7 e la corrispondente figura 12.8.

Listato 12.7.

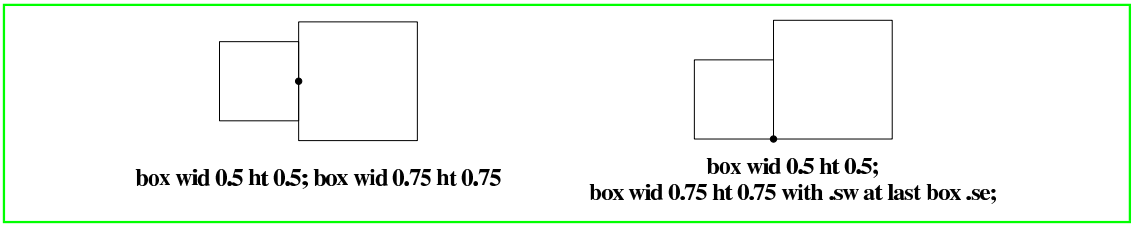
```
box "partenza"
move 0.75;
ellipse "arrivo"
arc -> cw from 1/3 of the way \
    between last box .n and last box .ne to last ellipse .n;
```

Figura 12.8. Una connessione un po' complicata, descritta con una sintassi naturale.



Il modifcatore '**`with`**' consente di identificare punti di due oggetti. Si tratta di un modo molto naturale per connettere gli oggetti. Si consideri, a mo' d'esempio, la figura 12.9.

Figura 12.9. Uso del modificatore '**with**' (sono evidenziati i punti di giunzione).



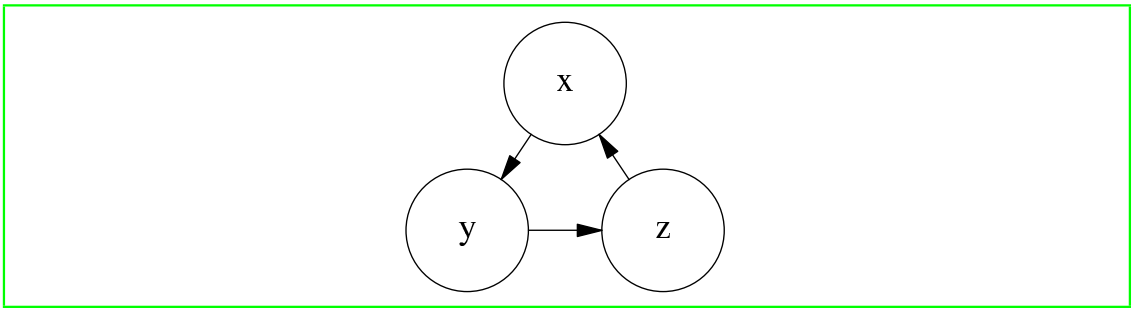
12.5 Il modificatore «chop»

Quando le linee tracciate fra due cerchi non li intersecano nei punti cardinali, è utile poterle accorciarle della lunghezza dei raggi dei cerchi ad entrambi i capi. Il listato 12.10 e la corrispondente figura 12.11 illustrano l'utlilizzo del modificatore '**chop**'.

Listato 12.10.

```
.PS
circle "x"
circle "y" at 1st circle - (0.4, 0.6)
circle "z" at 1st circle + (0.4, -0.6)
arrow from 1st circle to 2nd circle chop
arrow from 2nd circle to 3rd circle chop
arrow from 3rd circle to 1st circle chop
.PE
```

Figura 12.11. Il modificatore '**chop**'.



Si osservi che il modificatore '**chop**' sposta le punte di freccia, non le tronca. Per difetto, il modificatore '**chop**' accorcia entrambi i capi della linea di una lunghezza pari a '**circlerad**'. Aggiungendo un suffisso numerico è possibile modificare tale lunghezza.

Scrivendo:

```
line ... chop primo_raggio chop secondo_raggio
```

con ***primo_raggio*** e ***secondo_raggio*** valori numerici, è possibile modificare la lunghezza di accorciamento ad entrambi i capi. Usando tale descrizione assieme alle funzioni trigonometriche, è possibile scrivere codice che gestisce intersezioni ancora più complicate.

[1] Espresse i pollici.

[2] Per la precisione, possono essere addizionate una posizione e una coppia di espressioni; quest'ultima deve comparire come secondo termine dell'operazione.

[3] Un possibile utilizzo è per posizionare correttamente delle didascalie.

[4] I valori **non** sono ristretti all'intervallo [0..1].

Gruppi di oggetti

Esistono due diverse maniere per raggruppare più oggetti in Pic:

- ***raggruppamento a graffe*** (*brace grouping*);
- ***blocco composto*** (*block composite*).

13.1 Raggruppamenti a graffe

Il metodo più semplice per raggruppare un insieme di oggetti è mediante una coppia di parentesi graffe (‘**{**’ e ‘**}**’). All’uscita dal raggruppamento la posizione corrente e la direzione corrente vengono reimpostate ai loro valori precedenti all’ingresso nel raggruppamento.

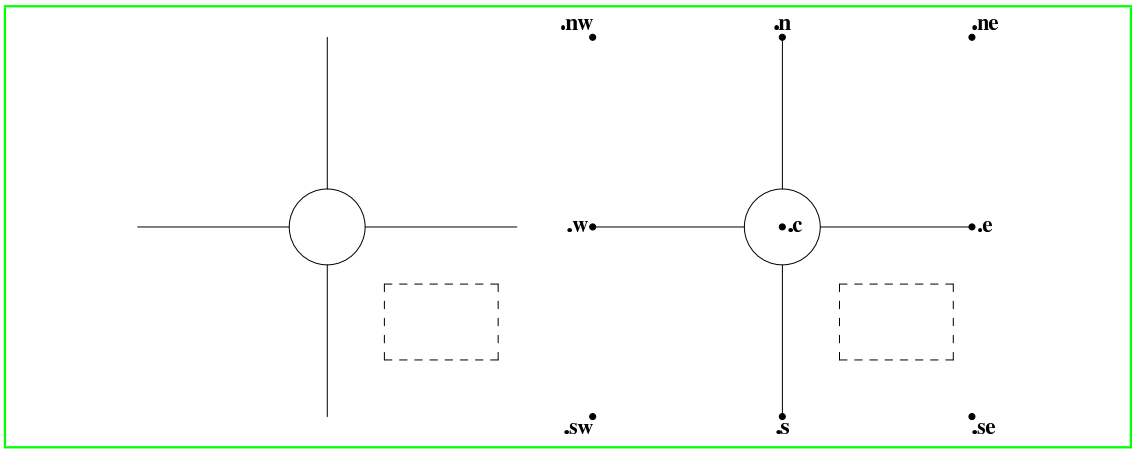
13.2 Blocchi composti

Un ***blocco composto*** è un oggetto creato da una successione di comandi racchiusa fra parentesi quadre (‘**[**’ e ‘**]**’). Il ***blocco composto*** è analogo per molti aspetti a un singolo oggetto chiuso, avente la forma e le dimensioni del riquadro circoscritto. Ad esempio, il frammento di codice del listato 13.1 corrisponde alla figura 13.2 (la quale mostra il blocco senza e con punti di giunzione.). L’etichetta ‘**A**’ corrisponde alla posizione del blocco.

Listato 13.1.

```
...
A: [
    circle;
    line up 1 at last circle .n;
    line down 1 at last circle .s;
    line right 1 at last circle .e;
    line left 1 at last circle .w;
    box dashed with .nw at last circle .se + (0.2, -0.2);
    Didascalia: center of last box;
]
...
```

Figura 13.2. Esempio di blocco composto.

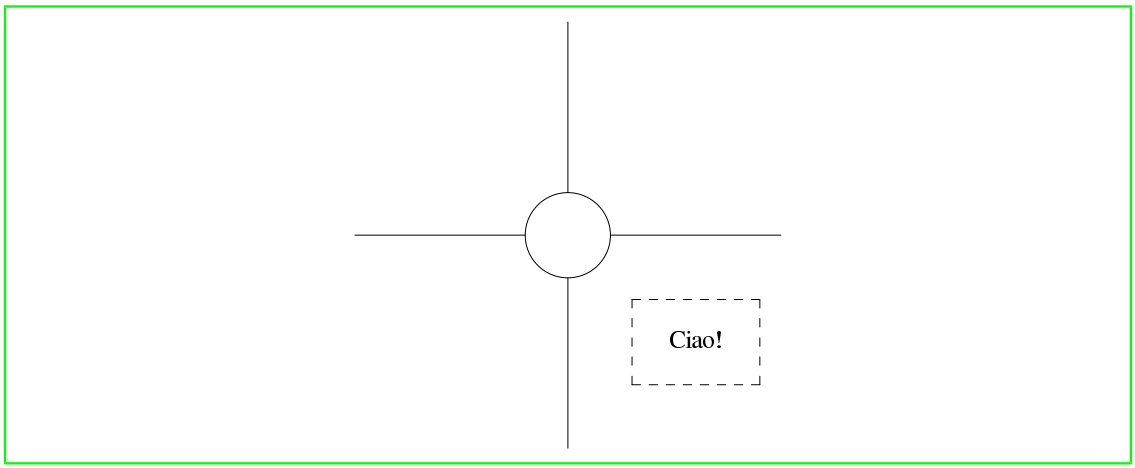


Per riferirsi a uno dei punti di giunzione del blocco composto, è possibile usare, ad esempio, la descrizione '**`A .s`**'. Ai fini della loro menzione, i blocchi composti sono delle ***classi***. Si può utilizzare la descrizione '**`last [] .s`**' equivalentemente, ovunque sia necessaria l'indicazione di una posizione. Tale costrutto è molto importante al fine di assemblare diagrammi estesi e complessi.

I blocchi composti forniscono anche un meccanismo per il campo di azione delle variabili, analogamente a un ***ambiente*** Troff (v. sezione 23). Gli assegnamenti di variabile effettuati all'interno di un blocco vengono annullati all'uscita. Per accedere a un valore interno a un blocco, si deve scrivere il nome del blocco, seguito da un punto, seguito dall'etichetta desiderata. Ad esempio, si potrebbe far riferimento al centro del riquadro nel blocco composto, di cui a listato 13.1 e figura 13.2, mediante '**`last [] .Didascalia`**' oppure '**`A.Didascalia`**'.

Questo tipo di riferimento a un'etichetta può essere utilizzato esattamente come una qualsiasi posizione. Ad esempio, aggiungendo '**`"Ciao!" at A.Didascalia`**' al listato 13.1 si ottiene la figura 13.3.

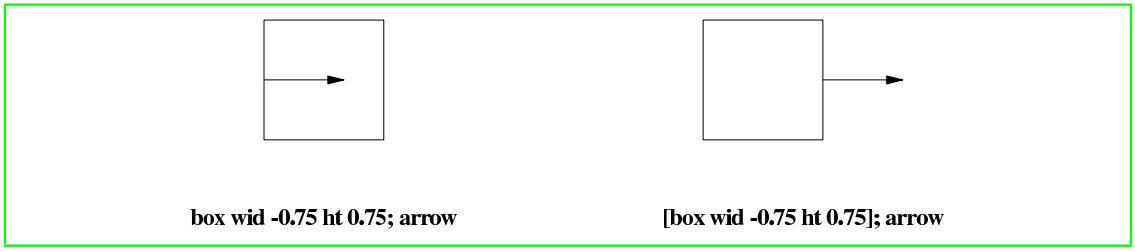
Figura 13.3. Aggiunta di una didascalia mediante un'etichetta interna.



È altresì possibile utilizzare le etichette interne sia a destra che a sinistra del modificatore '**`with`**'. Ciò significa che l'esempio di cui a listato 13.1 e figura 13.2 si potrebbe posizionare relativamente alla posizione del riquadro delle didascalia mediante un comando contentente '**`with A.Didascalia at`**'.

Si osservi che larghezza e altezza dei blocchi composti devono sempre essere considerate positive (figura 13.4).

Figura 13.4. I blocchi composti hanno sempre larghezza e altezza positive.



I blocchi composti si possono annidare; ciò significa che si possono utilizzare i punti di giunzione dei blocchi per costruire complesse strutture gerarchiche, dall'interno all'esterno. Si osservi che '**`last`**' e gli altri meccanismi di denominazione sequenziale non scendono ai livelli più interni, sicché nel listato 13.5 la freccia di cui alla riga 7 partirà dall'oggetto '**`P`**' e **non** dall'oggetto '**`Q`**'.

Listato 13.5. Blocchi composti annidati.

```
1  .PS
2  P: [box "foo"; ellipse "bar"];
3  Q: [
4       [box "baz"; ellipse "quxx"]
5       "testo assurdo";
6     ]
7  arrow from 2nd last [];
8  .PE
```

DWB Pic prevedeva che ci si potesse riferire ai livelli interni sino al massimo di un livello; GNU Gpic ha eliminato tale restrizione.

Variabili di stile

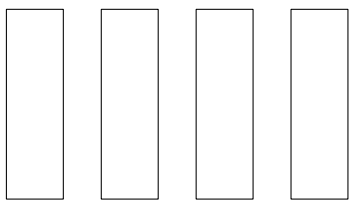
Esistono diverse variabili di stile globali in Pic, le quali possono essere utilizzate per variare il comportamento generale. Alcune di esse sono già state in precedenza menzionate; in questa sezione vengono presentate integralmente, specificandone anche il valore predefinito (tabella 14.1).

Tabella 14.1. Variabili di stile.

Variabile di stile	Valore predefinito (in pollici)	Che cosa determina
`moveht`	0,5	Ampiezza di uno spostamento vericale
`movewid`	0,75	Ampiezza di uno spostamento orizzontale
`lineht`	0,5	Ampiezza verticale di una linea (o spline) o di un segmento di spezzata
`linewid`	0,5	Ampiezza orizzontale di una linea (o spline) o di un segmento di spezzata
`textht`	0	Altezza del riquadro circoscritto a un oggetto di testo
`textwid`	0	Larghezza del riquadro circoscritto a un oggetto di testo
`arrowht`	0,1	Ampiezza longitudinale delle punte di freccia
`arrowwid`	0,05	Ampiezza trasversale delle punte di freccia
`arrowhead`	1	Abilita/disabilita campitura delle punte di freccia
`dashwid`	0,05	Intervallo di tratteggio per le linee
`maxpswid`	11	Larghezza massima della figura
`maxpsht`	8,5	Altezza massima della figura
`scale`	1	Fattore di scala
`fillval`	0,5	Valore di campitura

Tutte queste variabili possono essere impostate con una semplice istruzione di assegnamento (esempio in figura 14.2).

Figura 14.2.



boxht=1; boxwid=0.3; movewid=0.2; box; move; box; move; box; move; box;

In GNU Gpic impostando la variabile '**`scale`**' si scalano tutte le variabili di stile (dimensio-

nali) in modo tale che i loro valori producano risultati equivalenti anche nella nuova scala. Per verificarlo, si consideri la seguente sessione interattiva (vedi anche sezione 17.2):

```
$ cat x.pic [ Invio ]

  .PS
  print boxwid
  oldboxwid=boxwid
  scale=2.54
  print boxwid
  print oldboxwid*scale
  .PE


$ cat x.pic | pic > /dev/null [ Invio ]

  0.75
  1.905
  1.905


$
```

Il comando '**reset**' reimposta tutte le variabili di stile al valore predefinito. Fornendo una lista di nomi di variabili come argomenti (opzionalmente separati da virgole) si possono selettivamente reimpostare solo alcune variabili. Lo stato delle variabili di stile persiste da figura a figura.

Espressioni, variabili e assegnamenti

Un numero è ovviamente un'espressione valida.[1] La notazione decimale è legittima; in GNU Gpic è anche legittima la notazione scientifica nello stile del linguaggio C (come '**5e-2**').

Ovunque si può utilizzare un numero, il linguaggio accetta anche una variabile. Le variabili possono essere variabili di stile predefinite (vedi sezione 14), oppure nuove variabili create da un assegnamento.

DWB Pic prevede solamente l'assegnamento ordinario mediante il simbolo di uguaglianza ('**=**'), il quale definisce la variabile (a primo membro) nel blocco corrente, a meno che non sia già stata ivi definita, e poi ne cambia il valore (che si trova a secondo membro) nel blocco corrente. La variabile non è visibile al di fuori del blocco.[2]

GNU Gpic prevede anche un tipo di assegnamento alternativo, mediante l'operatore '**:=**'. In tal caso la variabile **deve essere già stata definita**, e il valore viene assegnato alla variabile senza creare una variabile locale al blocco corrente. Si consideri la seguente sessione interattiva (vedi anche sezione 17.2):

```
$ cat x.pic [ Invio ]

  .PS
  x=5
  y=5
  [
    x:=3
    y=3
  ]
  print x " " y
  .PE


$ cat x.pic | pic > /dev/null [ Invio ]

  3 5


$
```

Nelle espressioni si possono utilizzare l'altezza, la larghezza, il raggio e le coordinate cartesiane di qualsiasi oggetto o vertice. Se '**A**' è un'etichetta associata a un oggetto, le espressioni elencate nel listato 15.3 sono tutte valide.

Listato 15.3. Espressioni valide a partire dall'etichetta '**A**'.

```
1  ...
2  A.x                 # ascissa del centro di A
3  A.ne.y              # ordinata del vertice nordest di A
4  A.wid               # larghezza di A
5  A.ht                # la sua altezza
6  2nd last circle.rad # raggio del penultimo cerchio
7  ...
```

Si noti in particolare l'espressione in riga 3 del listato 15.3, la quale mostra come estrarre le coordinate di un vertice.

Sono disponibili le espressioni aritmetiche fondamentali, con una sintassi simile al linguaggio C: ‘**+**’, ‘*****’, ‘**-**’, ‘**/**’ e ‘**%**’. Inoltre c’è ‘**^**’ per l’elevamento a potenza. Si possono associare i termini in modo tradizionale mediante parentesi. GNU Gpic permette inoltre l’uso di operatori logici e di confronto: ‘**!**’, ‘**&&**’, ‘**||**’, ‘**==**’, ‘**!=**’, ‘**>=**’, ‘**<=**’, ‘**>**’, ‘**<**’.

Sono previste diverse funzioni predefinite: ‘**sin(*x*)**’, ‘**cos(*x*)**’, ‘**log(*x*)**’, ‘**exp(*x*)**’, ‘**sqrt(*x*)**’, ‘**max(*x*,*y*)**’, ‘**atan2(*x*,*y*)**’, ‘**min(*x*,*y*)**’, ‘**int(*x*)**’, ‘**rand()**’ e ‘**srand()**’. Sia ‘**exp**’ che ‘**log**’ si intendono in base 10; ‘**int**’ produce il troncamento all’intero precedente; ‘**rand**’ restituisce un numero pseudocasuale nell’intervallo [0..1) mentre ‘**srand**’[3] imposta il seme per una nuova successione di valori restituiti da ‘**rand**’.

GNU Gpic prevede anche una funzione ad un argomento ‘**rand(*x*)**’, che restituisce un numero pseudocasuale nell’intervallo [0..***x***), ma si tratta di una funzione deprecata che potrebbe venir rimossa nelle future versioni del programma.

La funzione ‘**sprintf**’ è simile all’omonima funzione della libreria standard del linguaggio C, ma prevede solamente le metavariabili di formato ‘**%**’, ‘**%e**’, ‘**%f**’ e ‘**%g**’.

[1] Tutti i numeri sono conservati internamente come in virgola mobile

[2] La situazione è simile al linguaggio C, in cui una variabile in un blocco impedisce di accedere alla variabile omonima esterna al blocco.

[3] Estensione GNU.

Macro

Pic offre la possibilità di definire delle ***macro***. La cosa può tornar utile per descrivere diagrammi ripetitivi. Assieme ale regole di visibilità per i blocchi composti (sezione 13.2) consente effettivamente di scrivere delle funzioni (vedi ad esempio il listato 16.1).

La sintassi è la seguente:

```
define nome_macro separatore corpo_macro separatore
```

In questo modo si definisce ***nome_macro*** come una macro che poi verrà sostituita dal testo in ***corpo_macro*** (separatori esclusi).[1] La macro può essere invocata come segue:

```
nome_macro(primo_argomento, secondo_primo_argomento, ...)
```

Gli (eventuali) argomenti vengono sostituiti al posto degli elementi sintattici ***$1***, ***$2***, ... ***$n*** che appaiono nel corpo della macro.

Come esempio di utilizzo delle macro, si considerino il listato 16.1 e la figura 16.2.

Listato 16.1. Esempio di utilizzo delle macro.

```
.PS
# Traccia un microinterruttore in un riquadro,
# $1 rappresenta lo stato (acceso/spento).
define jumper { [
    shrinkfactor = 0.8;
    Outer: box invis wid 0.45 ht 1;


    # Grazie al ] finale le seguenti variabili verranno
    # poi reimpostate automaticamente
    boxwid = Outer.wid * shrinkfactor / 2;
    boxht  = Outer.ht  * shrinkfactor / 2;


    box fill (!$1) with .s at center of Outer;
    box fill ($1)  with .n at center of Outer;
] }


# Traccia un blocco di sei microinterruttori
define jumperblock { [
    right;

    jumper($1);
    jumper($2);
    jumper($3);
    jumper($4);
    jumper($5);
    jumper($6);
```

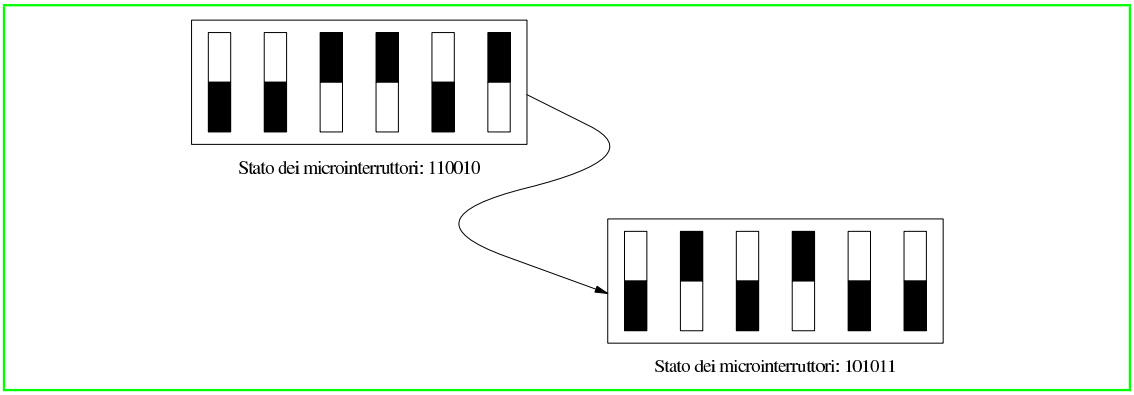
```
    jwidth  = last [].Outer.wid;
    jheight = last [].Outer.ht;


    box with .nw at 6th last [].nw wid 6*jwidth ht jheight;


    "Stato dei microinterruttori: $1$2$3$4$5$6" at last box .s + (0, -0.2);
] }


# Esempio di invocazione macro.
A: jumperblock(1,1,0,0,1,0);
move right 2 then down 1;
B: jumperblock(1,0,1,0,1,1);
spline -> from A.right down right 1 then down left 2 to B.left
.PE
```

Figura 16.2. Esempio di utilizzo delle macro.



Un dettaglio che l'esempio del listato 16.1 non illustra è che il passaggio dei parametri alle macro non comporta la separazione delle stringhe: chiamando '**`jumper( 1 )`**', il valore di ***$1*** sarà '**`1`**'; chiamando '**`jumper(stringa lunga)`**' ***$1*** varrà '**`stringa lunga`**'.
Se si intende passare come parametro una coppia ordinata, per evitare conflitti con il simbolo di separazione degli argomenti è possibile avvolgere la coppia fra parentesi.

La definizione di una macro è persistente da una figura all'altra. Per annullare una definizione si scriva:

```
undef nome_macro;
```

per esempio, il codice del listato 16.3 annulla le due macro definite nel listato 16.1.

Listato 16.3.

```
...
undef jumper
undef jumperblock
...
```


[1] ***separatore*** può essere qualunque carattere che non compaia in ***corpo_macro***, oppure la coppia ***separatore***/***separatore*** può essere costituita da una coppia bilanciata di parentesi graffe (‘**{**’ e ‘**}**’), e (in ogni caso) ***corpo_macro*** può contenere coppie bilanciate di parentesi graffe.

Sezione 17

Importazione ed esportazione dei dati

In questa sezione vengono descritti i comandi Pic che permettono lo scambio di dati fra Pic e l'ambiente esteno.

17.1 Inclusione di file e dati tabulati

Mediante la scrittura seguente:

```
copy nome_file
```

si inserisce il contenuto del file ***nome_file*** nel flusso di input di Pic. Le coppie di macro '**.PS**'/'**.PE**' eventualmente presenti vengono ignorate; in tal modo si possono includere figure precostruite.

Una variante[1] è la seguente:

```
copy [nome_file] thru {nome_macro | {separatore corpo_macro separatore}}
```

mediante la quale viene invocata la macro ***nome_macro*** - o eseguito il codice ***corpo_macro*** indicato in modo letterale - sugli argomenti ottenuti spezzando ciascuna riga del file ***nome_file*** in campi separati da spazi vuoti. La macro può avere sino a nove argomenti. Il codice ***corpo_macro*** dev'essere incluso in parentesi graffe, o fra una coppia di separatori (che però non possono comparire in ***corpo_macro***).

Se si omette di indicare il file, le righe da elaborare vengono prelevate dal resto del flusso di input di Pic, sino all'occorrenza successiva della macro '**.PE**'.

In ogni caso, assieme al comando '**copy**' GNU Gpic consente l'utilizzo di un suffisso '**until** ***parola***', la cui aggiunta permette di controllare la conclusione del comando '**copy**'[2].

Di conseguenza, i listati 17.1 e 17.2 si equivalgono.

Listato 17.1. '**copy**' con '**until**'.

```
.PS
copy thru % circle at ($1,$2) % until "FINE"
1 2
3 4
5 6
FINE
box
.PE
```

Listato 17.2. '**copy**' senza '**until**'.

```
.PS
circle at (1,2)
circle at (3,4)
circle at (5,6)
box
.PE
```

17.2 Messaggi di *debug*

Il comando '**print**' accetta un numero qualunque di argomenti, li concatena nella loro forma di output e invia il risultato verso lo standard error. Ciascun argomento può essere un'espressione, una posizione oppure una stringa di testo.

Per degli esempi elementari, si considerino le sezioni 14 e 15.

17.3 Invio di comandi al postprocessore

La seguente scrittura:

```
command argomento ...
```

fa sì che Pic concateni gli argomenti e li invii a Troff.[3] Ciascun ***argomento*** dev'essere un'espressione, una posizione o del testo.

Il funzionamento di '**command**' è simile alle righe inizianti con '**.**' (sezione 19) oppure con '****' (sezione 20), con la differenza che permette di passare i valori delle variabili.

17.4 Esecuzione di comandi della shell

La scrittura seguente:

```
sh separatore testo_qualsiasi separatore
```

espande le macro eventualmante presenti in ***testo_qualsiasi***, dopodiché lo esegue come comando della shell.[4] Tale funzionalità può essere utilizzata per generare figure o file di dati per una successiva rielaborazione. I separatori possono essere una coppia bilanciata di parentesi graffe, oppure dei caratteri che non siano presenti in ***testo_qualsiasi***.[5]

[1] Mututata dal linguaggio Grap.

[2] In pratica, il comportamento predefinito equivale a ‘**`until .PE`**’.

[3] Oppure a TeX, se vengono utilizzate le opzioni ‘**`-t`**’ o ‘**`-c`**’.

[4] Per difetto, GNU Gpic non esegue in realtà il comando, per motivi di sicurezza. Per forzare l’esecuzione bisogna invocare il programma con l’opzione ‘**`-U`**’.

[5] In entrambi i casi, ***testo_qualsiasi*** può contenere coppie bilanciate di parentesi graffe, e le eventuali stringhe possono contenere anche graffe non bilanciate.

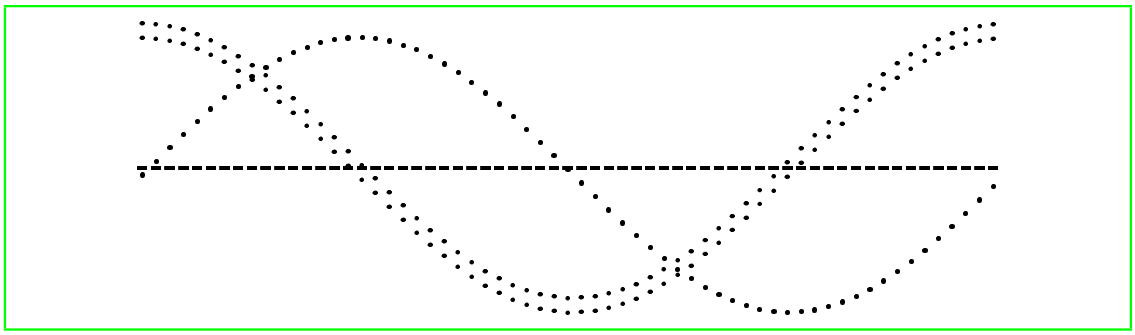
Controllo del flusso

Il linguaggio Pic prevede ***istruzioni condizionali*** e ***cicli enumerativi*** (listato 18.1 e figura 18.2).

Listato 18.1. Esempio di ciclo enumerativo.

```
.PS
pi = atan2(0, -1);
for i = 0 to 2 * pi by 0.1 do {
    "-" at (i/2, 0);
    "." at (i/2, sin(i)/2);
    ":" at (i/2, cos(i)/2);
}
.PE
```

Figura 18.2. Grafico realizzato mediante un ciclo enumerativo.



Ecco la sintassi per il ciclo enumerativo:

```
for variabile = prima_espressione to seconda_espressione [by [*]terza_espressione] ↩
↪do separatore corpo separatore
```

Ed eccone la semantica:

> Impostare ***variabile*** al valore di ***prima_espressione***. Fintantoché il valore di ***variabile*** non supera il valore di ***seconda_espressione***, eseguire ***corpo*** e incrementare ***variabile*** di una quantità pari a ***terza_espressione***; in assenza della parola chiave '**by**', incrementare ***variabile*** di una unità. Se ***terza_espressione*** è preceduta dal simbolo '*****' allora ***variabile*** deve venire moltiplicata per (**e non incrementata di**) una quantità pari a ***terza_espressione***.[1]

Ecco la sintassi per l'istruzione condizionale:

```
if espressione then separatore se_vero separatore [else altro_separatore se_falso altro_separatore]
```

Ed eccone la semantica:

> Valutare ***espressione***; se non è nulla allora eseguire ***se_vero***, altrimenti eseguire ***se_falso***.[2]

Nelle espressioni condizionali possono comparire gli usuali operatori relazionali: '**!**', '**&&**', '**||**', '**==**', '**!=**', '**>=**', '**<=**', '**>**', '**<**'.

È altresì previsto il confronto fra stringhe, mediante '**==**' e '**!=**'.

I confronti fra stringhe vanno inseriti fra parentesi per evitare ambiguità sintattiche.

[1] ***separatore*** può essere qualsiasi carattere che non compaia in ***corpo*** (o, alternativamente, la coppia ***separatore***/***separatore*** può essere una coppia bilanciata di parentesi graffe, come nel caso del comando '**sh**', v. sezione 17.4).

[2] ***separatore*** può essere qualsiasi carattere che non compaia in ***se_vero***. ***altro_separatore*** può essere qualsiasi carattere che non compaia in ***se_falso***. Entrambe le coppie di separatori possono in alternativa essere una coppia bilanciata di parentesi graffe, come nel caso del comando '**sh**' (v. sezione 17.4). In ogni caso sia ***se_vero*** sia ***se_falso*** possono contenere coppie bilanciate di parentesi graffe. Il bilanciamento non è richiesto per parentesi graffe presenti nelle stringhe di testo.

Interfaccia verso *roff

L'output di Pic è costituito da comandi grafici per *roff.[1]

19.1 Argomenti per variare la scala

DWB Pic accetta uno oppure due argomenti di seguito alla macro '**.PS**'; essi vengono interpretati come larghezza e altezza[2] a cui va scalato il risultato dell'elaborazione di Pic.

GNU Gpic è meno flessibile: accetta solamente la larghezza finale a cui scalare l'immagine, oppure uno zero seguito dall'altezza massima a cui scalare; con due argomenti non nulli scalerà all'altezza massima.

19.2 Gestione della variazione di scala

Quando Pic elabora una descrizione proveniente dall'input, poi passa '**.PS**' e '**.PE**' al postprocessore. A '**.PS**' vengono aggiunti un paio di valori numerici, ossia le dimensioni della figura.[3] Il postprocessore utilizza tali valori per riservare abbastanza spazio per la figura e centrarla.

La realizzazione GNU del pacchetto di macro '**s**', ad esempio, include la definizione di cui al listato 19.1.

Listato 19.1. La definizione della macro '**.PS**' nel pacchetto di macro Groff '**s**'.

```
.\" ***************************
.\" ******** module pic ********
.\" ***************************
.\" Pic support.
.\" PS height width
.de PS
.br
.sp \\n[DD]u
.ie \\n[.$]<2 .@error bad arguments to PS (not preprocessed with pic?)
.el \{\
.       ds@need (u;\\$1)+1v
.       in +(u;\\n[.l]-\\n[.i]-\\$2/2>?0)
.\}
.HTML-IMAGE
..
```

C'è una definizione equivalente nel pacchetto di macro '**pic**', utilizzabile mediante l'opzione '**-mpic**'; ciò permette di usare anche pacchetti diversi da '**s**'.

Se si utilizza '**.PF**' al posto di '**.PE**', Troff reimposta la posizione allo stato precedente all'inizio della figura.[4]

La scrittura seguente:

```
.PS <file
```

fa sì che il contenuto di ***file*** vada a sostituire la riga stessa.

Trattasi di una funzionalità deprecata. Si consiglia, invece, di utilizzare '**copy** ***file***' (v. sezione 17.1).

Per difetto, le righe in input che iniziano con un punto ('**.**') vengono passate al postprocessore, nella corrispondente posizione dell'output.

«Giocare» con le spaziature, sia orizzontali che verticali, è una probabile fonte di errori, mentre cambiare le dimensioni dei punti e le fonti di solito non comporta problemi. Analogamente è di solito innocuo un cambiamento di dimensioni o di fonte nelle stringhe, a patto di ripristinarle prima del termine della stringa.

Lo stato della campitura per *roff è persistente da figura a figura.

---

[1] GNU Gpic si fonda su estensioni di tipo grafico presenti in Groff ma non in Troff.

[2] In pollici. Le dimensioni sono indipendenti.

[3] In pollici.

[4] Kernighan (sezione 23) fa notare che la lettera «F» sta per *flyback*.

Interfaccia verso TeX

La modalità TeX è attivata dall'opzione '**`-t`**'. In tale modalità, Pic definisce, per ciascuna figura, una scatola verticale (v. sezione 23) denominata '**`\graph`**'. È responsabilità del codice TeX visualizzare effettivamente la scatola verticale, ad esempio con il comando '**`\centerline{\box\graph}`**'.

In effetti, poiché la scatola verticale ha altezza zero, il precedente comando produrrà uno spazio leggermente superiore sopra la figura rispetto a sotto; per evitarlo basta scrivere '**`\centerline{\raise 1em\box\graph}`**'.

Le righe che iniziano con il carattere barra obliqua inversa ('`\`') vengono passate intatte a TeX; viene automaticamente aggiunto un '**`%`**' alla fine di ciascuna riga al fine di evitare spazi indesiderati. Si può utilizzare tranquillamante tale funzionalità per cambiare le fonti, o per cambiare il valore di '**`\baselineskip`**'.

Qualsiasi altro utilizzo potrebbe generare risultati imprevisti. Usare con cautela!

Le righe che iniziano con un punto ('`.`') non subiscono nessuna elaborazione particolare.

La modalità TeX di Pic **non** traduce i cambiamenti di fonte e dimensione dei caratteri contenute nelle stringhe di testo!

Comandi obsoleti

GNU Gpic prevede il comando seguente:

```
plot espressione ["testo"]
```

Trattasi di un oggetto costruito utilizzando ***testo*** come stringa di formattazione (compatibile con '**`sprintf`**'), con argomento ***espressione***. Omettendo ***testo*** viene usata implicitamente la stringa '**`%g`**'. Si possono specificate degli attributi alla maniera di una normale stringa di testo.

Si presti particolare cautela alla stringa di formattazione usata, poiché Pic non fa controlli molto accurati al riguardo.
Il comando '**`plot`**' è deprecato; si consiglia di utilizzare '**`sprintf`**' al suo posto.

Sezione 22

Indicazioni operative per ottenere le immagini

Per effettuare delle prove con il linguaggio Pic è necessario poter ottenere in qualche modo concreto l'immagine finale e visualizzarla. A tale scopo si indicano qui alcune possibilità.

Si può ad esempio utilizzare una pipeline del genere:

```
cat sorgente_pic | pic | troff -Tps | grops | ps2eps > file_eps
```

ottenendo quindi un file EPS, da visualizzare o ulteriormente elaborare. Addirittura si potrebbe preparare una riga di comando più complessa, la quale racchiuda in sé l'intero ciclo di modifica, compilazione e visualizzazione:

```
vi sorgente_pic; cat sorgente_pic | pic | troff -Tps | grops | ps2eps > file_eps; ↩
↪gv file_eps
```

Se la sintassi del sorgente Pic non è corretta, si ottiene un messaggio di errore standard, secondo lo schema seguente:

```
pic:sorgente_pic:numero_di_riga: syntax error before elemento
pic:sorgente_pic:numero_di_riga: giving up on this picture
```

ove ***elemento*** è un elemento sintattico, di solito immediatamente successivo alla causa dell'errore.

Un'alternativa sintatticamente più semplice prevede l'utlizzo di Pic2plot (parte del pacchetto GNU Plotutils):

```
cat sorgente_pic | pic2plot -Tps > file_eps
```

tenendo però presente che:

- Pic2plot riconosce solo un sottoinsieme del linguaggio Pic;
- Pic2plot non riconosce le sequenze di escape che iniziano con '**\(**', pertanto se si desiderano ottenere le lettere accentate si devono utilizzare delle sequenze di escape che iniziano con '\';
- l'aspetto finale dell'immagine può essere leggermente differente rispetto a quanto ottenuto mediante Troff.

Un'ulteriore semplificazione si può ottenere utilizzando il programma di Eric S. Raymond '**pic2graph**', il quale si appoggia a Groff e perciò riconosce l'intero linguaggio Pic:

```
cat sorgente_pic | pic2graph > file_png
```

Va precisato che, purtroppo, al momento della stesura del presente lavoro il programma '**pic2graph**' non funziona correttamente, nel senso che il file prodotto non ha le dimensioni corrette ed è pertanto inutilizzabile.

Se si intende utilizzare del codice Eqn all'interno del codice Pic, si tenga presente che (secondo quanto dichiarato nella pagina di manuale[1]) la realizzazione GNU di Eqn (Geqn) non è del tutto compatibile con il Troff tradizionale, pertanto è necessario utilizzare Groff:

```
cat sorgente | groff -e -p | ps2eps > file_eps
```

Per poter utilizzare del codice Eqn all'interno del codice Pic è necessario che nel sorgente si specifichino i delimitatori per il codice Eqn, mediante le macro '**.EQ**' e '**.EN**', ad esempio:

```
.EQ
delim $$
.EN
.PS
arrow; box "$1 over H(z)$"; arrow
.PE
```

Infine, si noti che Alml (il sistema di composizione SGML di Daniele Giacomini) offre supporto - a partire dall'estate 2005 - per il linguaggio Pic.

[1] *«[...] The syntax is quite compatible with Unix eqn. The output of GNU eqn cannot be processed with Unix troff; it must be processed with GNU troff. [...]»*

Riferimenti

- B. W. Kernighan, *PIC - A Graphics Language for Typesetting (Revised User Manual)*, Bell Labs Computing Science Technical Report #116, maggio 1991

  ⟨*http://cm.bell-labs.com/cm/cs/cstr/116.ps.gz* ⟩

- Eric S. Raymond, *Making Pictures With GNU PIC*[1]

  ⟨*http://www.catb.org/~esr/writings/taoup/html/graphics/pic.ps* ⟩

- Dale Dougherty, Tim O'Reilly, *UNIX ® Text Processing*, Hayden Books, edizione per Internet «UTP Revival», 2004

  ⟨*http://home.alltel.net/kollar/utp/* ⟩

- Daniele Giacomini, *Appunti di informatica libera*, capitoli *Introduzione a *roff* e *TeX: paragrafi, righe, spazi, scatole e linee*

  ⟨*http://a2.swlibero.org/introduzione_a_roff.htm* ⟩

  ⟨*http://a2.swlibero.org/tex_paragrafi_righe_spazi_scatole_e_linee.htm* ⟩

[1] A questo lavoro si deve gran parte dei contenuti del presente capitolo.

Appendici

Appendice A

Informazioni aggiuntive sul software e altre opere citate

Massimo Piai *(... circa 1975)*
⟨*pxam67 (ad) virgilio·it* ⟩