

decksh reference



Keywords

Structure Text

deck
edeck
slide
eslide
canvas
include
grid
text
ctext
etext
rtext
arctext
textblock
textfile
textcode

Lists

list
blist
nlist
clist
li
elist

Graphics

acircle
arc
circle
curve
ellipse
hline
line
pill
polygon
rect
rrect
square
star
vline

Braces Arrows

lbrace
rbrace
ubrace
dbrace
arrow
crarrow
clarrow
cuarrow
cdarrow

Images

image
cimage

Charts

dchart
legend

Loop

for
efor

Assignments

polarx
polary
area
format
random
vmap

Data

data
edata

Keywords and arguments

keyword

arguments

mandatory

optional

text `"..string...."` `x` `y` `n`

`"font"` `"color"` `op`

text "hello, world" 80 50 2

hello, world

text "hello, world" 80 40 2 "serif"

hello, world

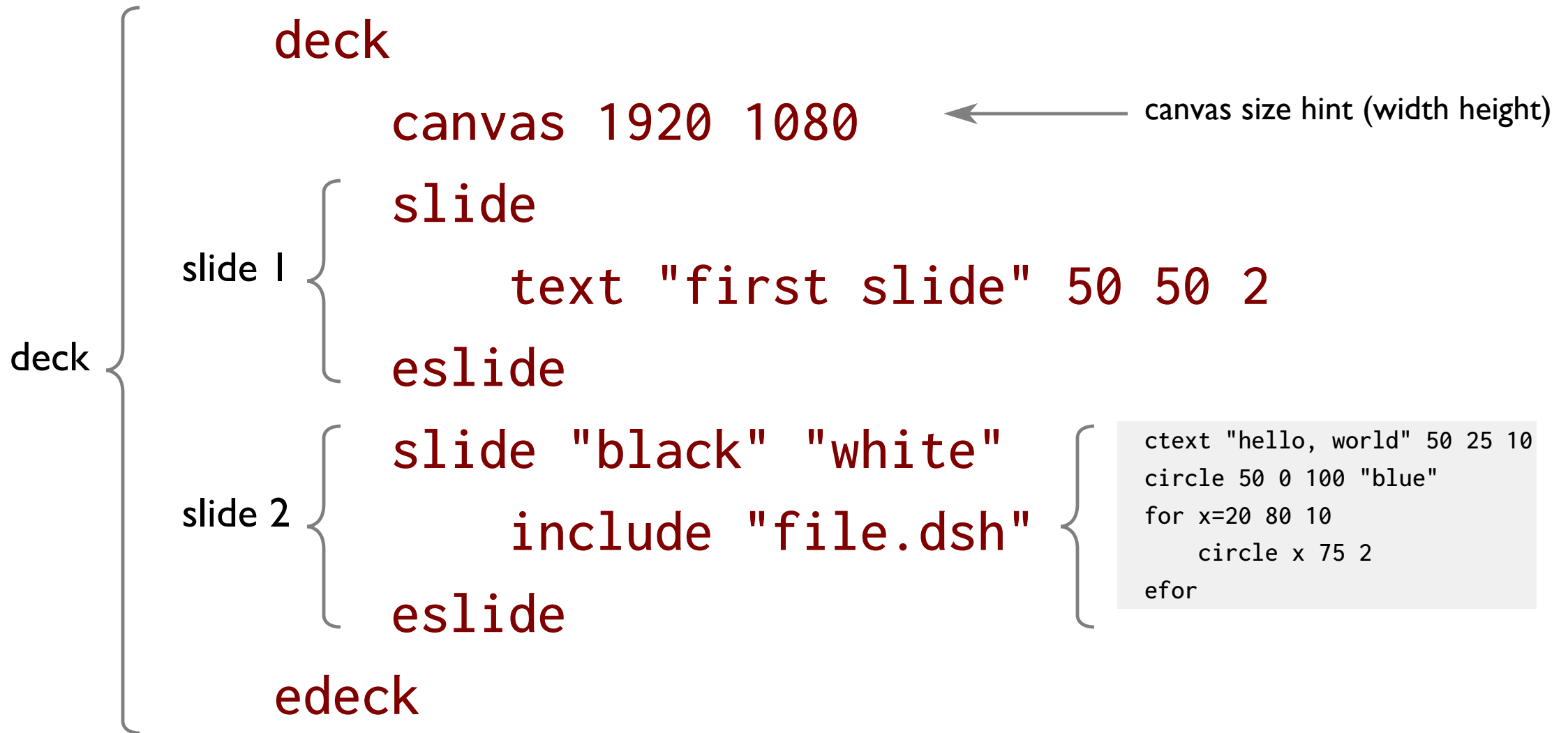
text "hello, world" 80 30 2 "serif" "red"

hello, world

text "hello, world" 80 20 2 "serif" "red" 50

hello, world

Structure



Percent Grid



Object Index

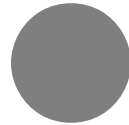
Text

hello, world

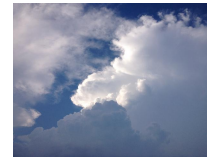
The quick brown
fox jumped over
the lazy dog

what's up, Doc?

Graphics



Images



sky

Lists

- First
- Second
- Third

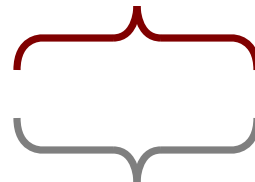
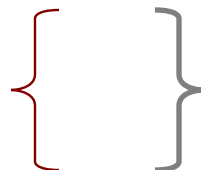
1. First
2. Second
3. Third

First
Second
Third

Arrows



Braces



Textual Elements

description	keyword	mandatory	optional
Left-aligned	text	"..." x y fontsize	font color op link
Centered	ctext	"..." x y fontsize	font color op link
End-aligned	etext	"..." x y fontsize	font color op link
Rotated	rtext	"..." x y angle fontsize	font color op link
Text on an arc	arctext	"..." x y rad a1 a2 fontsize	font color op link
Block text	textblock	"..." x y w fontsize	font color op link
File contents	textfile	"file" x y fontsize	font color op spacing
Code listing	textcode	"file" x y w fontsize	font color

hello, world
(x,y)

text x y fontsize font color op link

abc

```
text "abc" 20 20 4
```

abc

```
text "abc" 75 20 7 "mono" "maroon"
```


hello, world

(x,y)

`c`text x y fontsize font color op link

abc

```
c
```

text "abc" 20 20 4

abc

```
c
```

text "abc" 80 20 7 "mono" "maroon"

hello, world.

(x,y)

`etext x y fontsize font color op link`

abc

`etext "abc" 20 20 4`

abc

`etext "abc" 80 20 7 "mono" "maroon"`

hello, world

(x,y)

rttext x y angle fontsize font color op link

abc

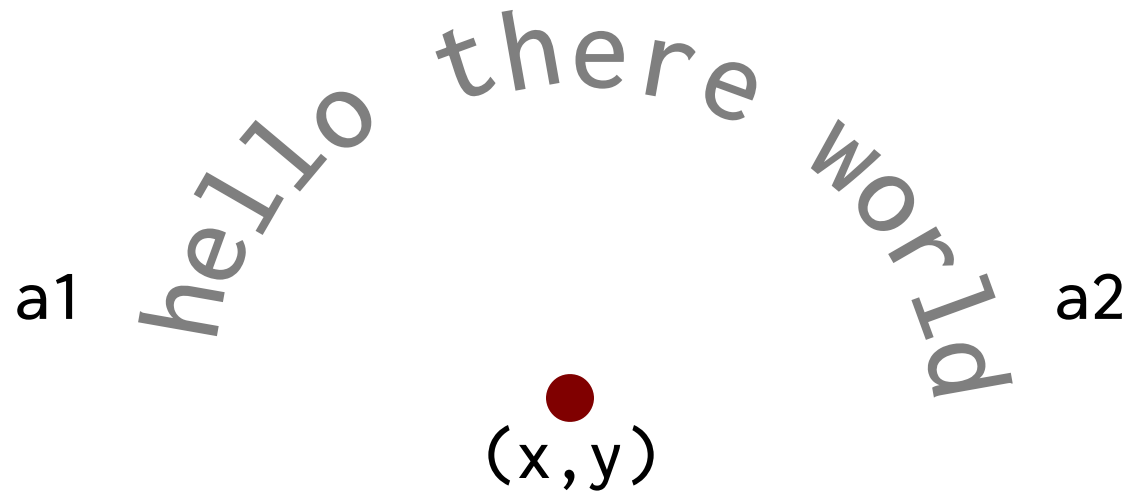
ctext 20 20 30 3

abc

ctext 50 20 90 5

abc

ctext 80 20 270 4 "sans" "maroon"



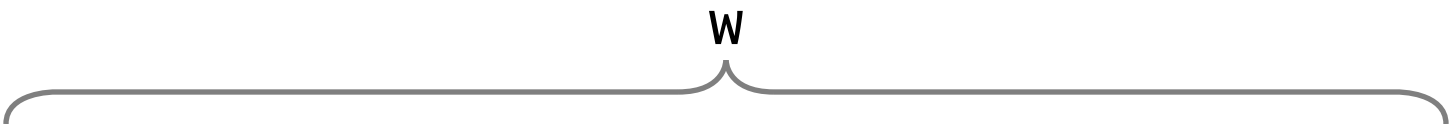
```
arctext x y radius a1 a2 fontsize font color op link
```

What is up

This is curvy

```
arctext "What is up" 25 20 10 180 90 3 "mono"
```

```
arctext "This is curvy" 75 30 10 180 360 3 "mono"
```

(x, y)  “Where justice is denied, where poverty is enforced, where ignorance prevails, and where any one class is made to feel that society is an organized conspiracy to oppress, rob and degrade them, neither persons nor property will be safe.”

textblock x y w fontsize font color op link

“Where justice is denied, where poverty is enforced, where ignorance prevails, and where any one class is made to feel that society is an organized conspiracy to oppress, rob and degrade them, neither persons nor property will be safe.”

textblock "... " 10 35 30 2

“Where justice is denied, where poverty is enforced, where ignorance prevails, and where any one class is made to feel that society is an organized conspiracy to oppress, rob and degrade them, neither persons nor property will be safe.”

textblock "... " 50 35 10 1 "sans" "maroon"

(x,y) This is the contents
of a file. it has lines of text.
Reading is fundamental.

`textfile` `x` `y` `fontsize` `font` `color` `op` `link`

This is the contents
of a file. it has lines of text.
Reading is fundamental.

`textfile "example.txt" 10 35 2`

```
package main
```

```
import "fmt"
```

```
func main() {
```

```
    fmt.Println("hello, world")
```

```
}
```

```
textfile "hw.go" 55 35 1.6 "mono" "maroon"
```

(x, y)

W

```
package main

import "fmt"

func main() {
    fmt.Println("hello, world")
}
```

textcode x y w fontsize font color

```
package main

import "fmt"

func main() {
    fmt.Println("hello, world")
}
```

textcode "hw.go" 10 35 25 1.0

```
package main

import "fmt"

func main() {
    fmt.Println("hello, world")
}
```

textcode "hw.go" 55 35 40 1.6 "maroon"

Graphical Elements

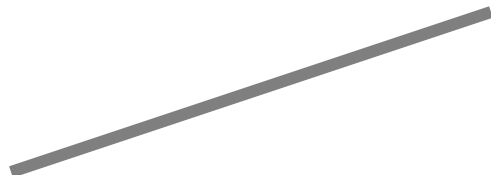
description	keyword	mandatory	optional
Line	line	x1 y1 x2 y2 lw	color op
Horizontal line	hline	x y w	lw color op
Vertical line	vline	x y h	lw color op
Elliptical arc	arc	x y w h a1 a2	lw color op
Quadratic Bezier	curve	bx by cx cy ex ey	lw color op
Circle	circle	x y w	color op
Area circle	acircle	x y area	color op
Ellipse	ellipse	x y w h	color op
Square	square	x y w	color op
Rectangle	rect	x y w h	color op
Rounded rectangle	rrect	x y w h radius [color]	color
Pill shape	pill	x y w h [color]	color
Polygon	polygon	"x1 x2...xn" "y1 y2...yn"	color op
N-sided star	star	x y sides inner outer	color op

lw { 
(x1,y1) (x2,y2)

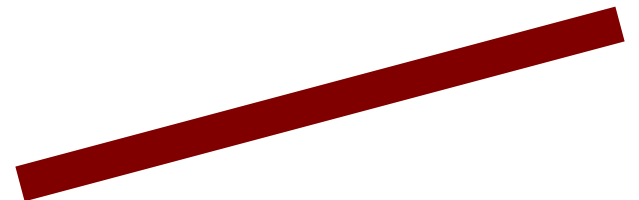
line x1 y1 x2 y2 lw color op



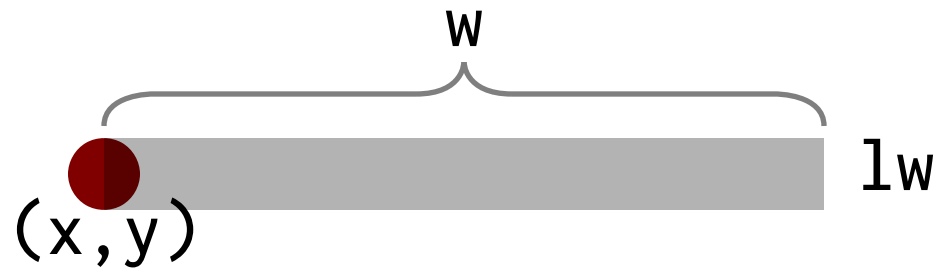
line 10 20 30 20



line 40 20 60 30 0.5



line 70 20 95 30 1.5 "maroon"



```
hline x y w lw color op
```



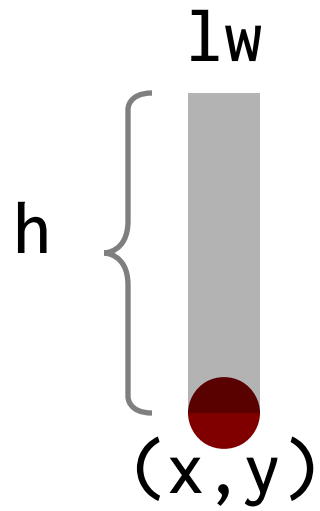
```
hline 15 20 10
```



```
hline 40 20 20 1
```



```
hline 70 20 20 5 "maroon" 20
```



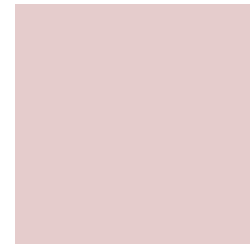
`vline` `x` `y` `w` `lw` `color` `op`



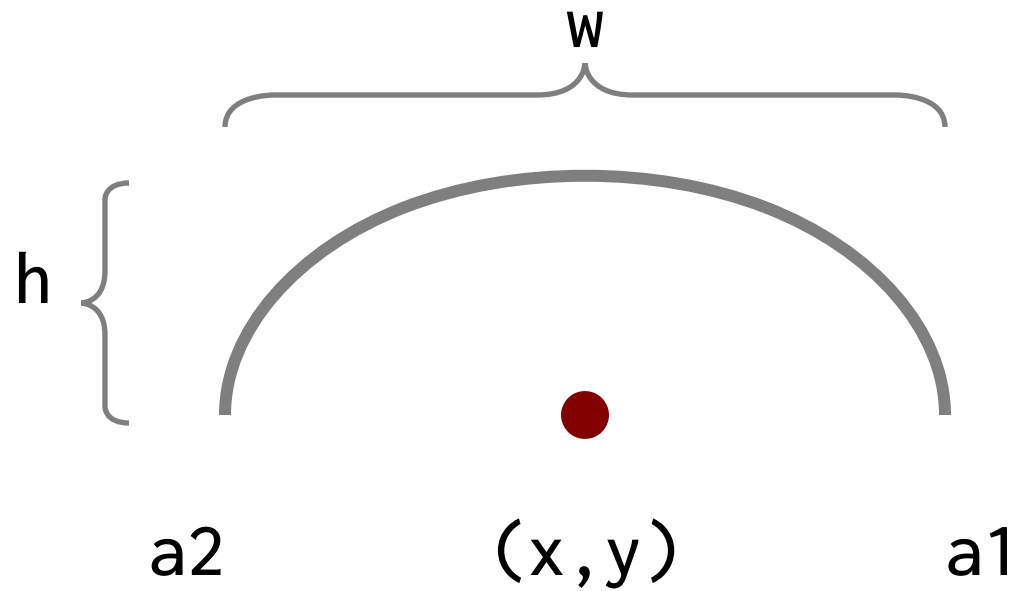
`vline 20 20 15`



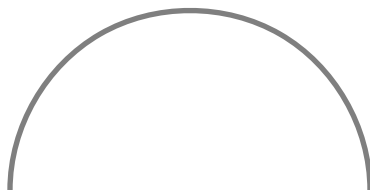
`vline 50 20 15 2`



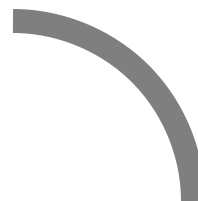
`vline 80 20 15 10 "maroon" 20`



`arc x y w h a1 a2 lw color op`



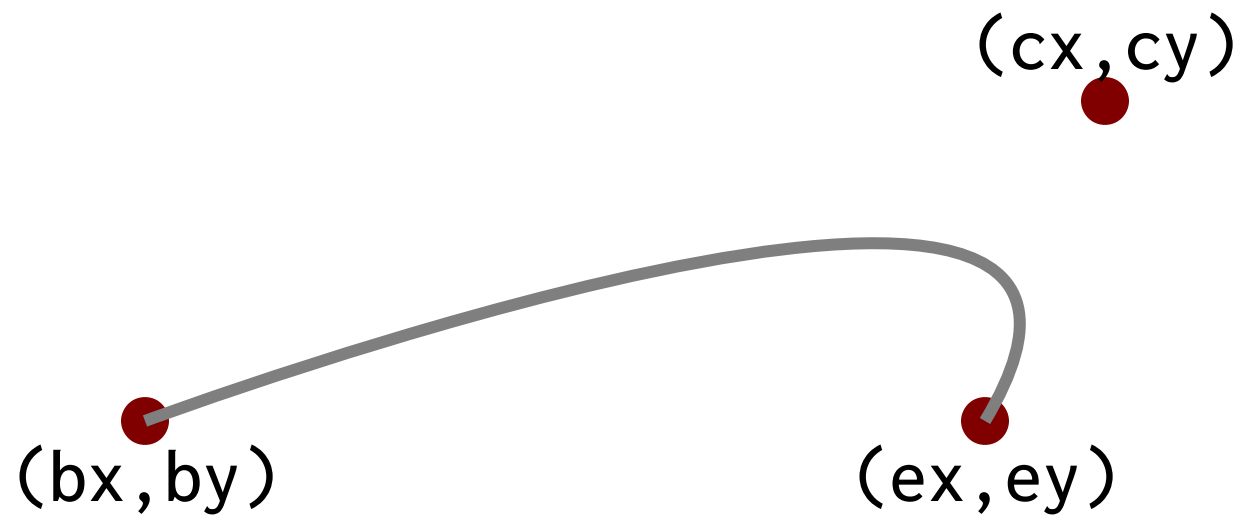
`arc 20 20 15 15 0 180`



`arc 50 20 15 15 0 90 1`



`arc 80 20 5 5 0 180 5 "maroon"`



`curve` `bx` `by` `cx` `cy` `ex` `ey` `lw` `color` `op`



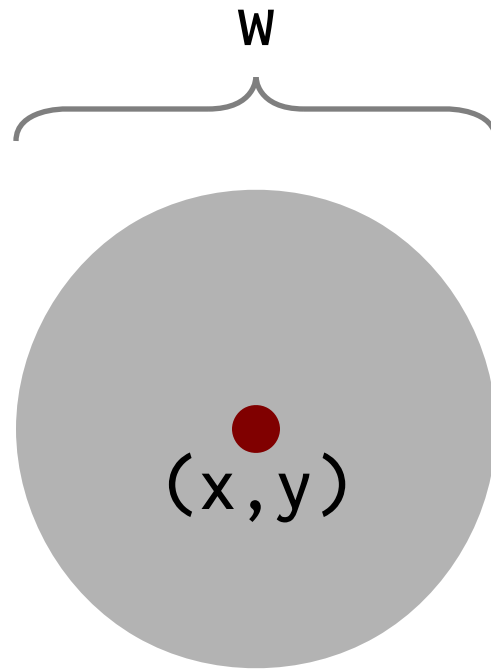
`curve 15 20 25 30 30 25`



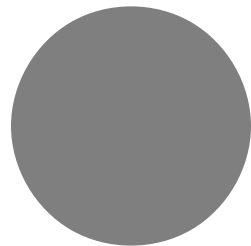
`curve 15 20 25 30 30 25`



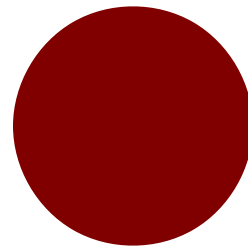
`curve 70 20 70 30 90 25 0.5 "maroon"`



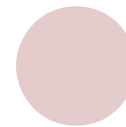
`circle x y w color op`



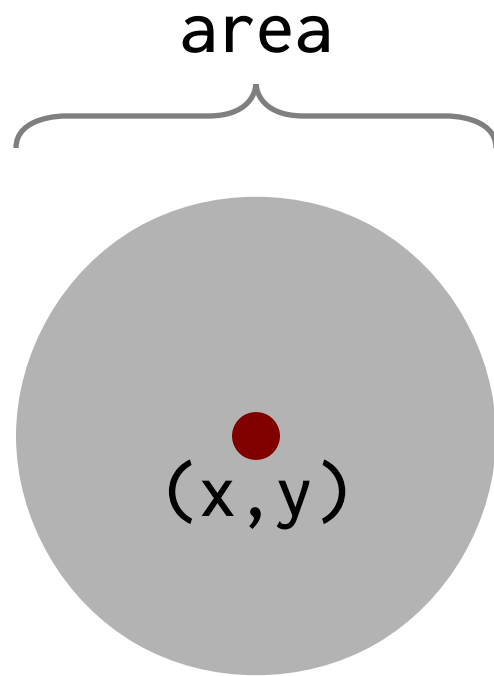
`circle 20 20 10`



`circle 50 20 10 "maroon"`



`circle 80 20 5 "maroon" 20`



`circle x y area color op`



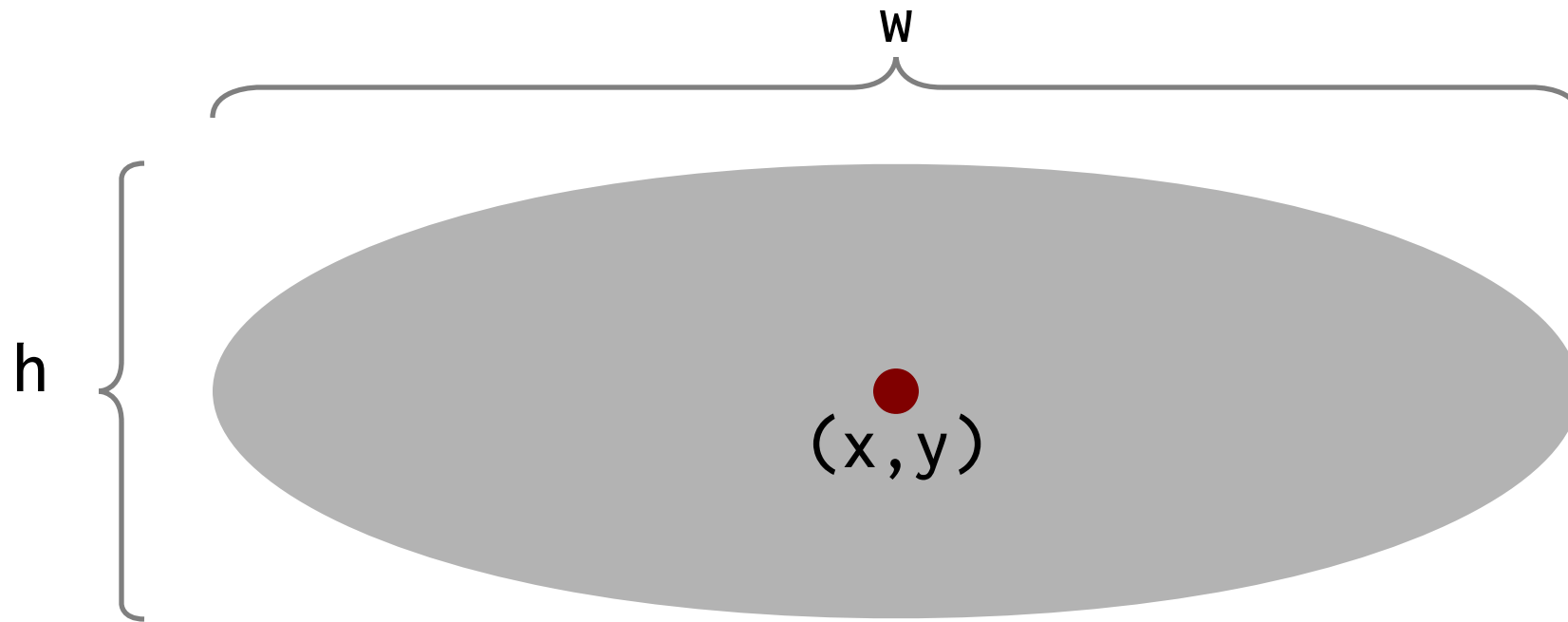
`acircle 20 20 10`



`acircle 50 20 10 "maroon"`



`acircle 80 20 5 "maroon" 20`



`ellipse x y w h color op`



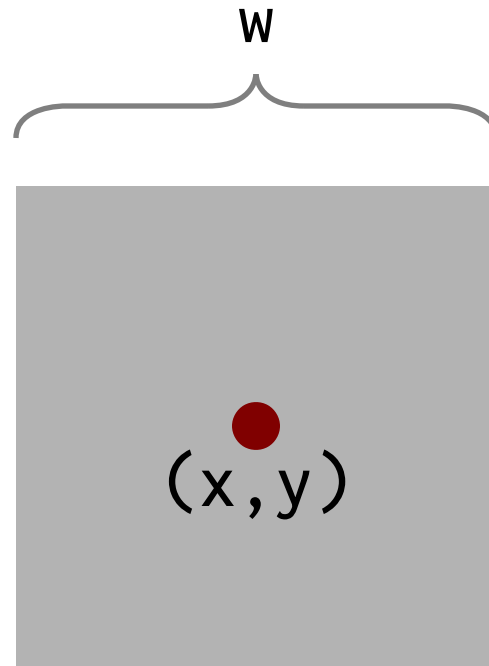
`ellipse 20 20 10 5`



`ellipse 50 20 10 5 "maroon"`



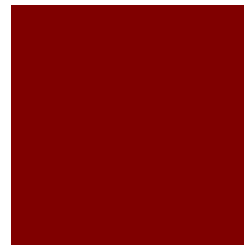
`ellipse 80 20 5 10 "maroon" 20`



`square x y w color op`



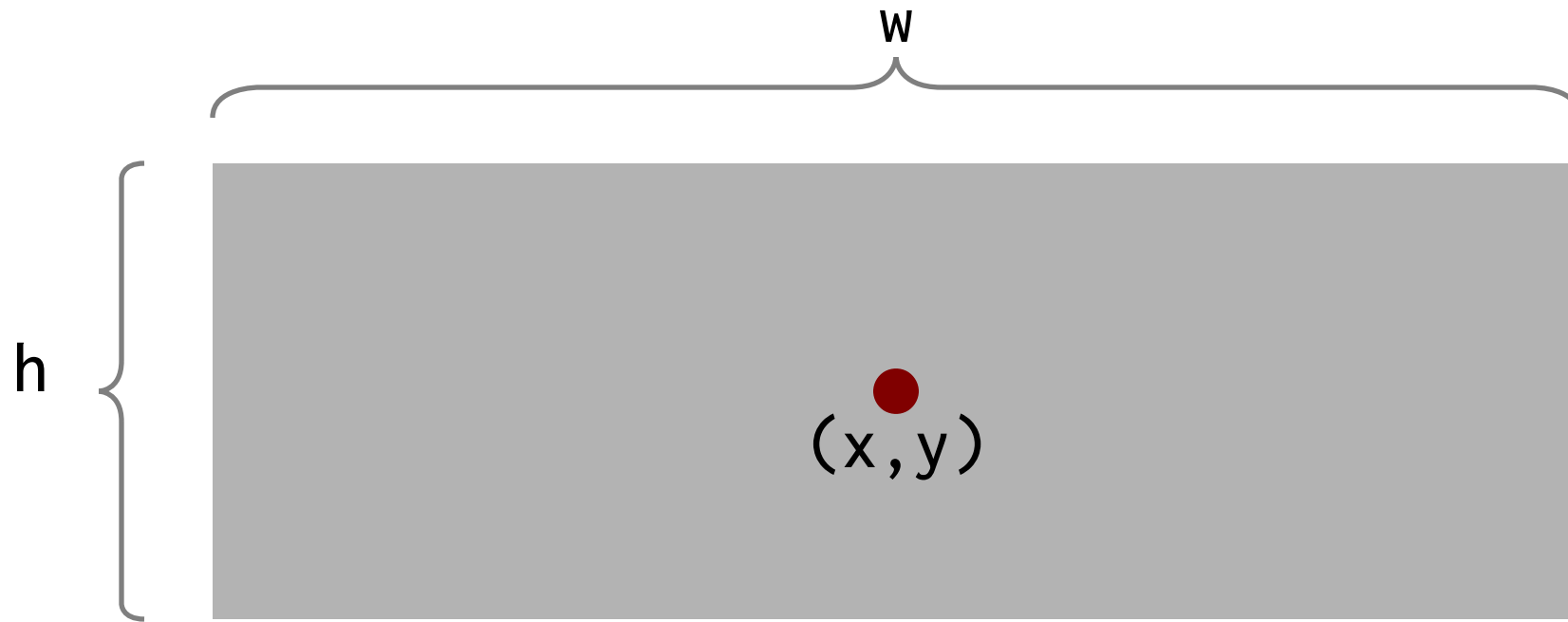
`square 20 20 10`



`square 50 20 10 "maroon"`



`square 80 20 5 "maroon" 20`



`rect x y w h color op`



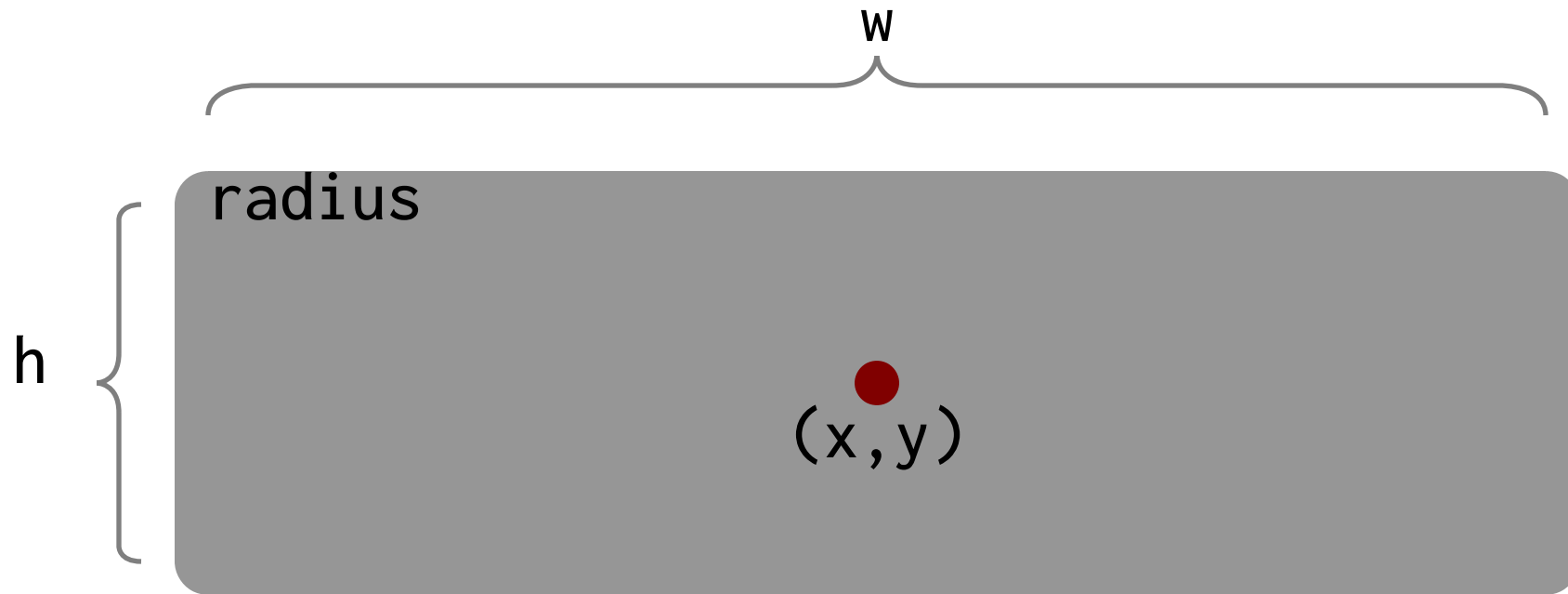
`rect 20 20 10 5`



`rect 50 20 10 5 "maroon"`



`rect 80 20 5 10 "maroon" 20`



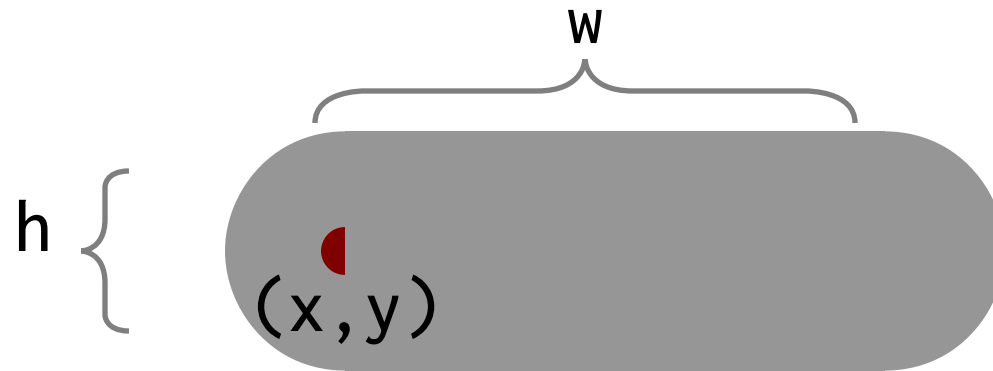
`rrect x y w h radius color op`



`rrect 20 20 10 5 1`



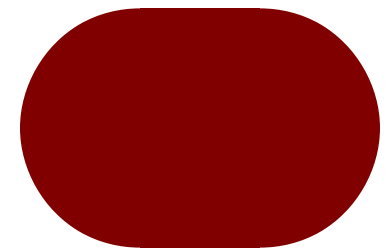
`rrect 80 20 5 10 1 "maroon"`



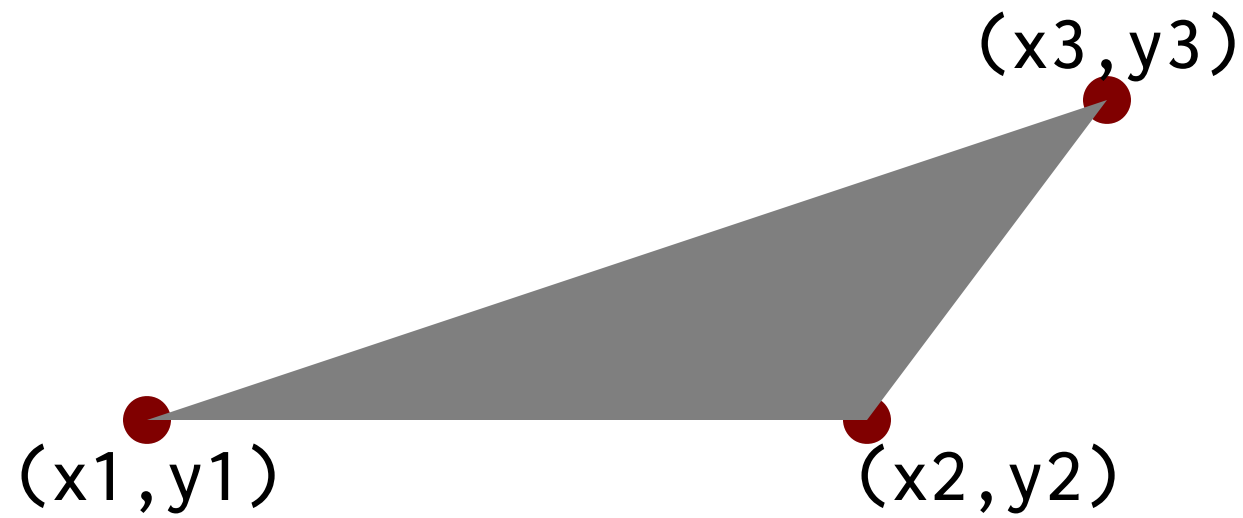
`pill x y w h color`



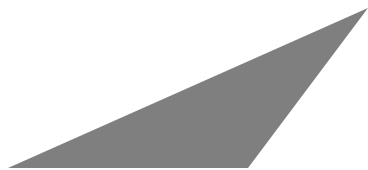
`pill 20 20 10 5`



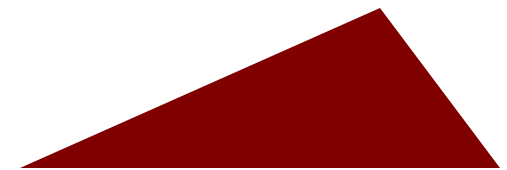
`pill 80 20 5 10 "maroon"`



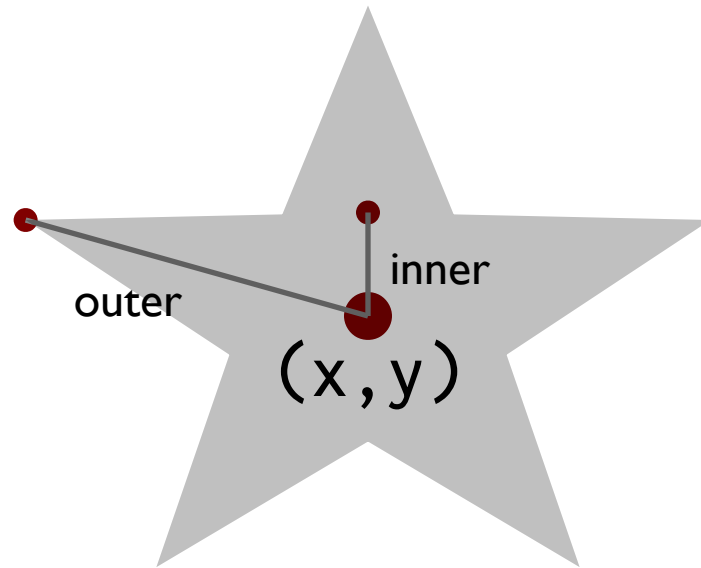
`polygon x1 x2...xn" "y1 y2...yn color op`



`polygon "10 25 20" "20 30 20"`



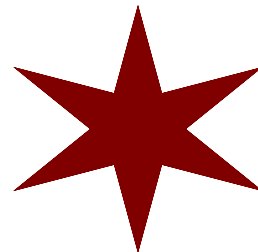
`polygon "70 85 90" "20 30 20" "maroon"`



`star x y sides inner outer color op`



`star 20 20 5 2 6`



`star 50 20 12 2 5 "maroon"`



`star 80 ey 24 2 8 "maroon" 20`

Images

description	keyword	mandatory	optional
Image	<code>image</code>	<code>"file" x y w h</code>	<code>scale "link"</code>
Captioned image	<code>cimage</code>	<code>"file" "caption" x y w h</code>	<code>scale "link"</code>

Note: the scale value is a percentage from 1-100, and link is a URL

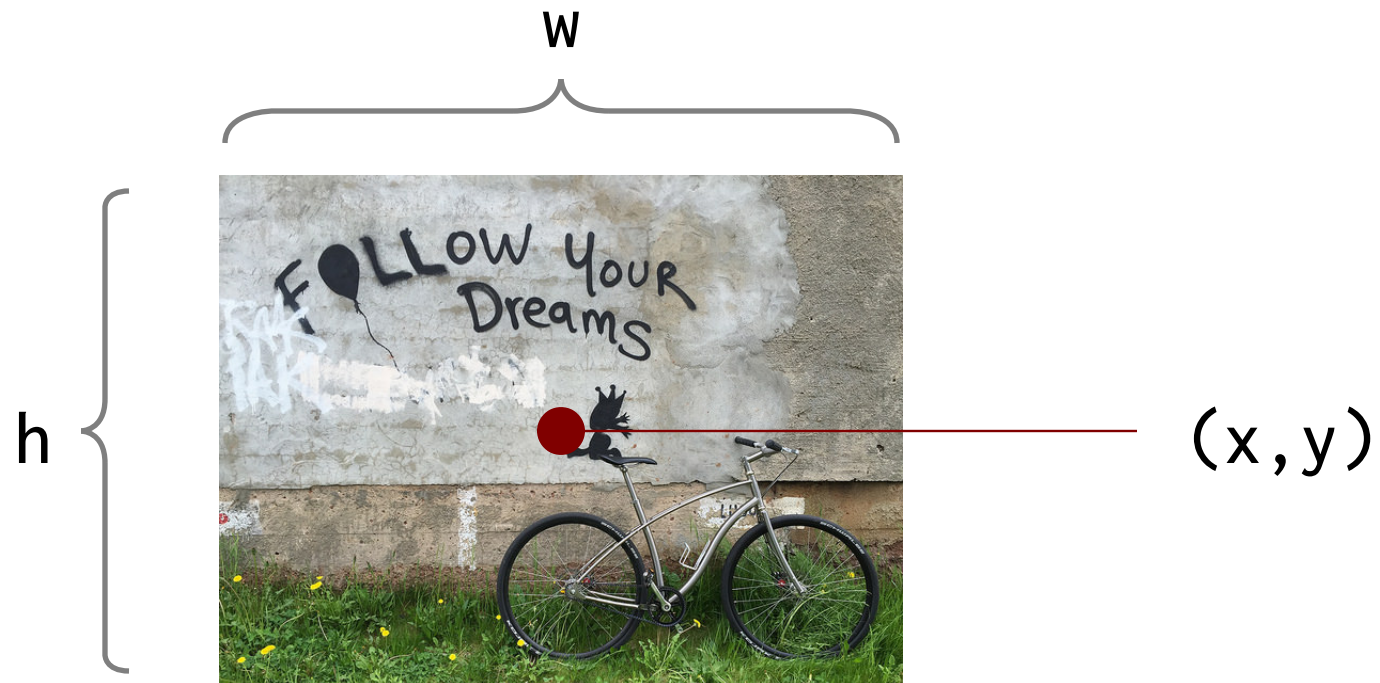


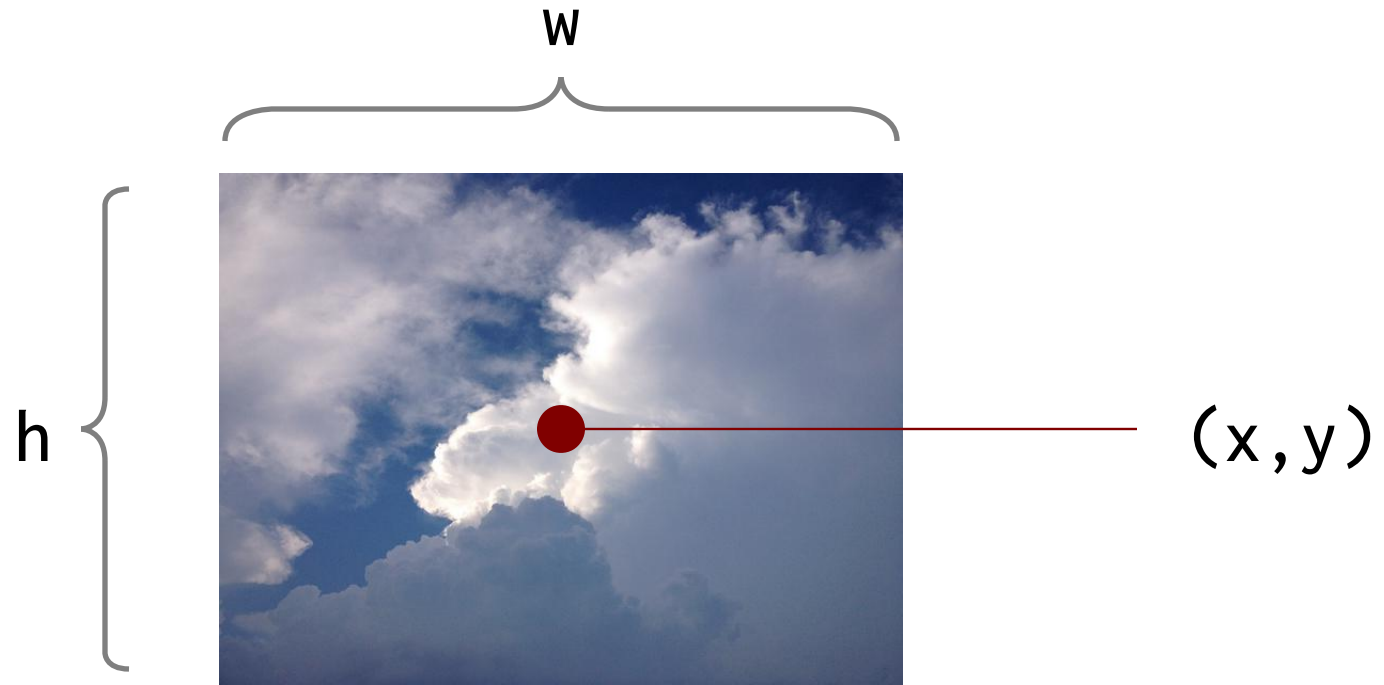
image x y w h scale link



image "follow.jpg" 20 25 640 480 10

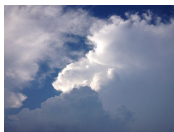


image "follow.jpg" 75 25 640 480 30



sky

`cimage x y w h scale link`



sky



sky

`cimage "cloudy.jpg" "sky" 20 25 640 480 10`

`cimage "cloudy.jpg" "sky" 75 25 640 480 30 "" 1.5`

Lists

description	keyword	mandatory	optional
Plain list	<code>list</code>	<code>x y fontsize</code>	<code>font color op spacing</code>
Bullet list	<code>blist</code>	<code>x y fontsize</code>	<code>font color op spacing</code>
Numbered list	<code>nlist</code>	<code>x y fontsize</code>	<code>font color op spacing</code>
Centered list	<code>clist</code>	<code>x y fontsize</code>	<code>font color op spacing</code>

```
list x y fs
(x,y) li "first"
      li "second"
      li "third"
elist
```

list x y fontsize font color op spacing

```
list 20 30 2.5 one
      li "one"
      li "two" two
      li "three" three
elist
```

```
list 85 30 4 "serif" "maroon" 100 1.0
      li "one"
      li "two"
      li "three"
elist
```

one
two
three

```
      ● blist x y fs
(x,y)  li "first"
        li "second"
        li "third"
      elist
```

blist x y fontsize font color op spacing

```
blist 20 30 2.5 ● one
  li "one"
  li "two" ● two
  li "three" ● three
elist
```

```
blist 85 30 4 "serif" "maroon" 100 1.0
  li "one"
  li "two"
  li "three"
elist
```

● *one*
● *two*
● *three*

```
nlist x y fs
(x,y) li "first"
      li "second"
      li "third"
elist
```

nlist x y fontsize font color op spacing

```
nlist 20 30 2.5 |. one
  li "one"
  li "two"      2. two
  li "three"    3. three
elist
```

```
nlist 85 30 4 "serif" "maroon" 100 1.0
  li "one"
  li "two"
  li "three"
elist
```

1. one
2. two
3. three

```
clist x y fs
(x,y) li "first"
      li "second"
      li "third"
elist
```

clist x y fontsize font color op spacing

```
clist 30 30 2.5
  li "first one"
  li "next"
  li "and last"
elist
```

first one
next
and last

```
clist 90 30 4 "serif" "maroon" 100 1.0
  li "first"
  li "next"
  li "and last"
elist
```

first
next
and last

Arrows

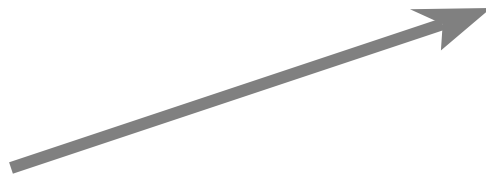
description	keyword	mandatory	optional
Straight	<code>arrow</code>	<code>x1 y1 x2 y2</code>	<code>lw aw ah color op</code>
Left curved	<code>lcarrow</code>	<code>bx by cx cy ex ey</code>	<code>lw aw ah color op</code>
Right curved	<code>rcarrow</code>	<code>bx by cx cy ex ey</code>	<code>lw aw ah color op</code>
Up curved	<code>ucarrow</code>	<code>bx by cx cy ex ey</code>	<code>lw aw ah color op</code>
Down curved	<code>dcarrow</code>	<code>bx by cx cy ex ey</code>	<code>lw aw ah color op</code>



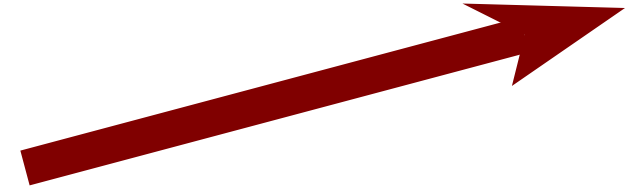
`arrow` `x1` `y1` `x2` `y2` `lw` `aw` `ah` `color` `op`



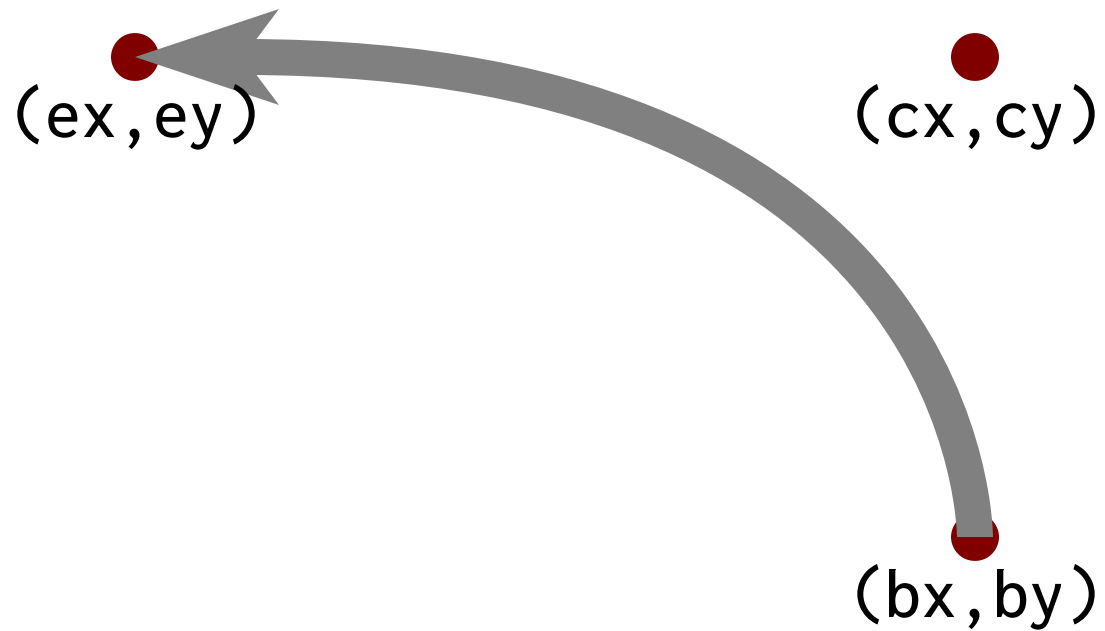
`arrow 10 20 30 20`



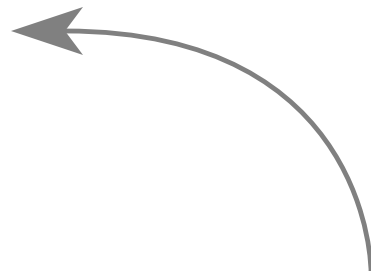
`arrow 40 20 60 30 0.5`



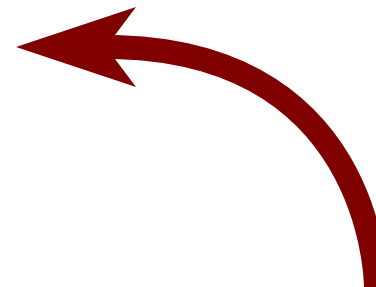
`arrow 70 20 95 30 1.5 6 6 "maroon"`



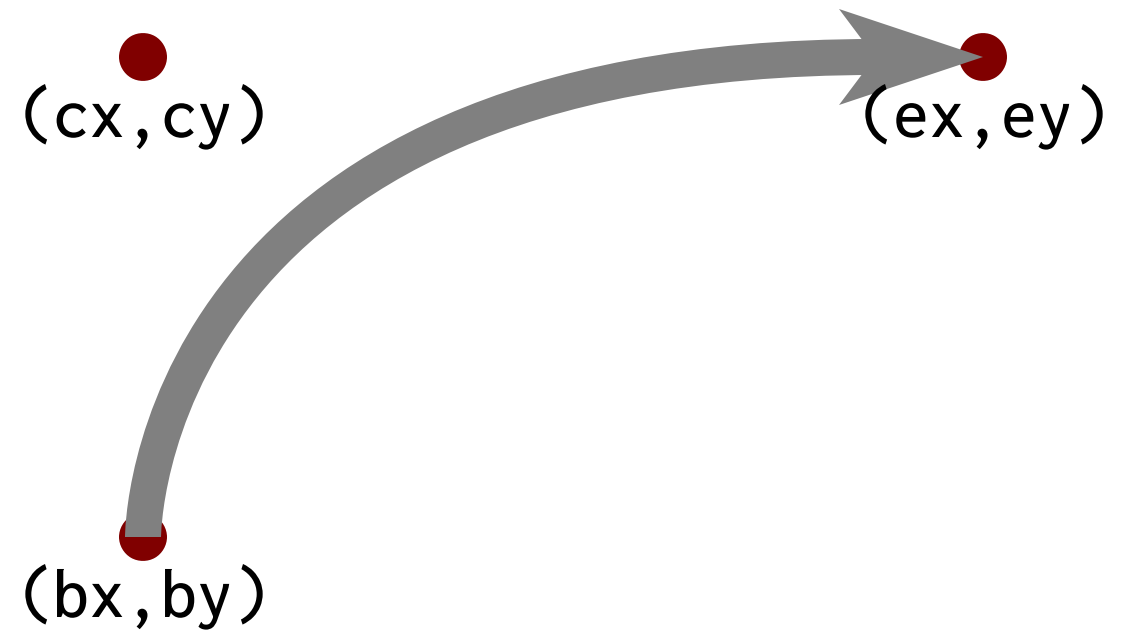
`lcarrow bx by cx cy ex ey lw aw ah color op`



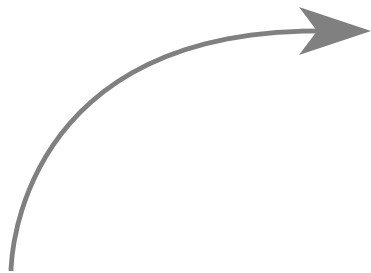
`lcarrow 30 20 30 35 15 35`



`lcarrow 70 20 70 35 55 35 1 5 5 "maroon"`



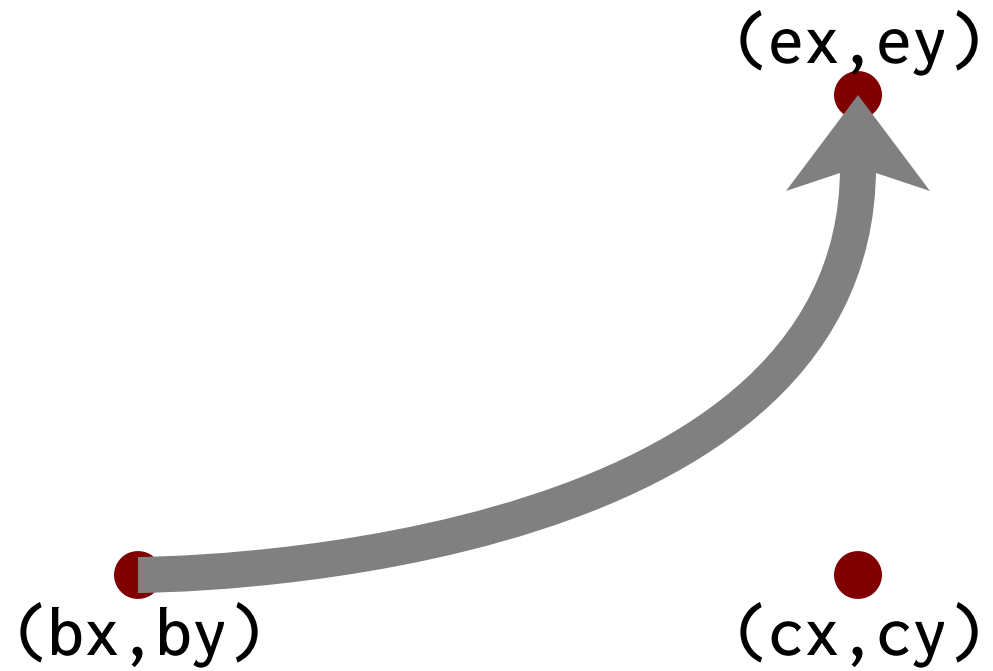
`rcarrow bx by cx cy ex ey lw aw ah color op`



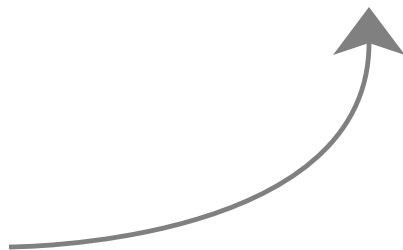
`rcarrow 15 20 15 35 30 35`



`rcarrow 50 20 50 35 70 35 1 5 5 "maroon"`



`ucarrow bx by cx cy ex ey lw aw ah color op`



`ucarrow 15 20 30 20 30 35`



`rcarrow 50 20 70 20 70 35 1 5 5 "maroon"`

(bx, by)

(cx, cy)

(ex, ey)

`dcarrow bx by cx cy ex ey lw aw ah color op`



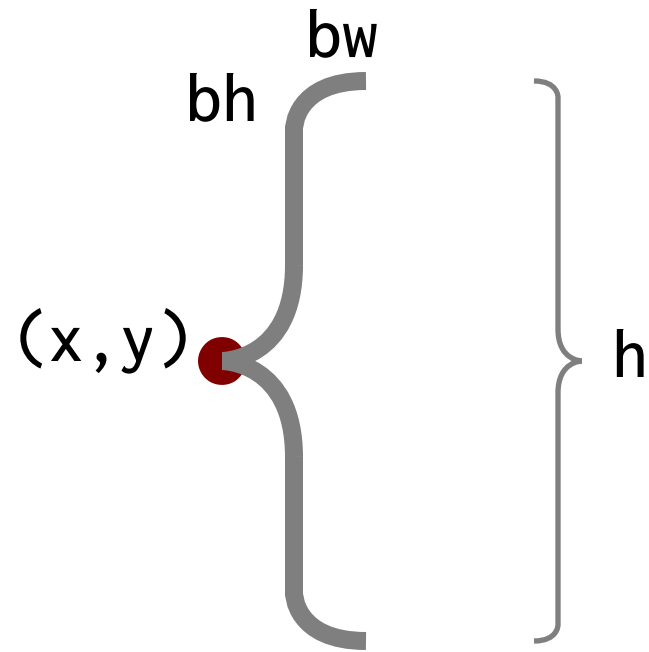
`dcarrow 15 35 30 30 20`



`dcarrow 50 35 70 35 70 20 1 5 5 "maroon"`

Braces

description	keyword	mandatory	optional
Left brace	lbrace	x y fontsize bw bh	lw color op
Right brace	rbrace	x y fontsize bw bh	lw color op
Up brace	ubrace	x y fontsize bw bh	lw color op
Down brace	dbrace	x y fontsize bw bh	lw color op



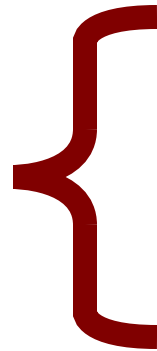
`lbrace` x y $fontsize$ bw bh lw $color$ op



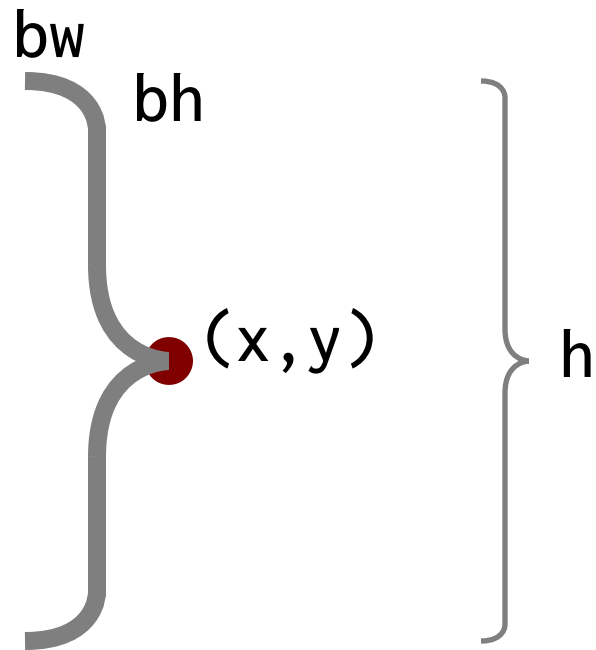
`lbrace 20 25 20 2 2`



`lbrace 50 25 20 4 4 1`



`lbrace 80 25 20 6 3 1 "maroon"`



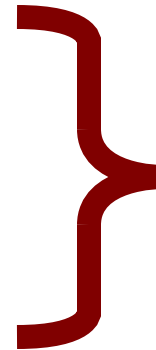
`rbrace` `x` `y` `fontsize` `bw` `bh` `lw` `color` `op`



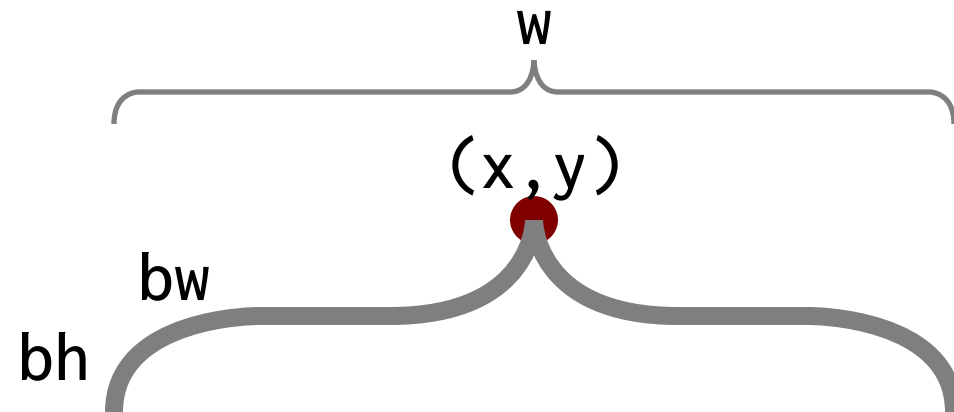
`rbrace 20 25 20 2 2`



`rbrace 50 25 20 4 4 1`



`rbrace 80 25 20 6 3 1 "maroon"`



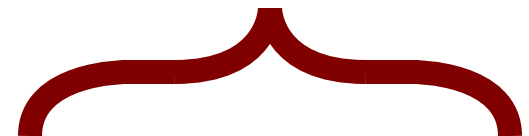
`ubrace x y fontsize bw bh lw color op`



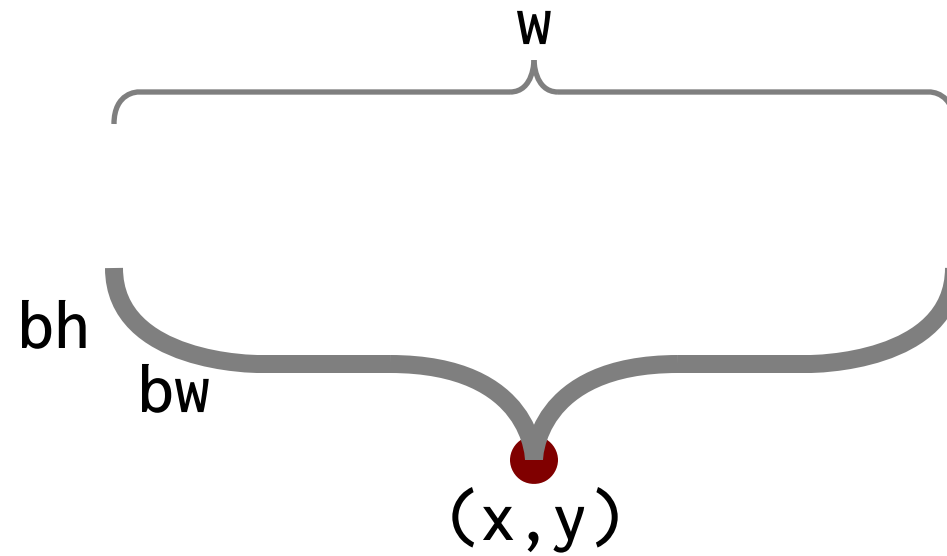
`ubrace 20 25 20 2 2`



`ubrace 50 25 20 4 4 1`



`ubrace 80 25 20 4 4 1 "maroon"`



dbrace x y fontsize bw bh lw color op



dbrace 20 25 20 2 2



dbrace 50 25 20 4 4 1



dbrace 80 25 20 4 4 1 "maroon"

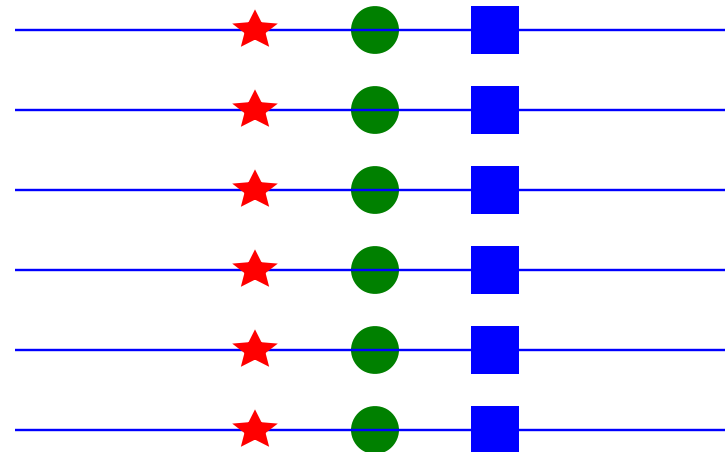
Loop, Assignments, Data and Grid

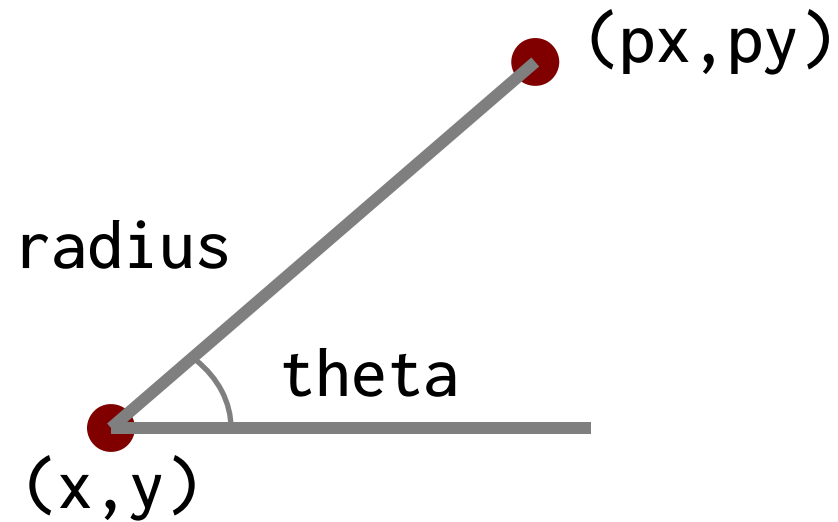
description	keyword	mandatory
Loop	<code>for v=</code>	<code>begin end [increment] ... efor</code>
Polar coordinate (x)	<code>x=polarx</code>	<code>x y radius angle</code>
Polar coordinate (y)	<code>y=polary</code>	<code>x y radius angle</code>
Area	<code>value=area</code>	<code>expression</code>
Formatted text	<code>value=format</code>	<code>fmt expression</code>
Random number	<code>value=random</code>	<code>min max</code>
Value mapping	<code>value=vmap</code>	<code>data min1 max1 min2 max2</code>
In-line data	<code>data</code>	<code>"file" ... edata</code>
Objects on a grid	<code>grid</code>	<code>grid "file" x y hspace vspace edge</code>

```
for v=begin end [increment]  
...items to repeat using v  
efor
```

```
for v=begin end increment ...efor
```

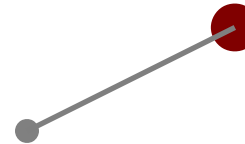
```
for v=10 35 5  
  hline 50 v 30 0.1 "blue"  
  star 60 v 5 1 0.4 "red"  
  circle 65 v 2 "green"  
  square 70 v 2 "blue"  
efor
```





```
px=polarx x y radius theta  
py=polary x y radius theta
```

```
cpx=60  
cpy=20  
px1=polarx cpx cpy 10 30  
py1=polary cpx cpy 10 30  
line cpx cpy px1 py1  
circle cpx cpy 1 "gray"  
circle px1 py1 2 "maroon"
```



v=123.45

a=area v



area



original value

value=**area** expression

m1=100

m2=200

a1=area m1

a2=area m2

circle 60 20 a1 "maroon"

circle 80 20 a2



x=3.14159

y=2.0

title=format "Value=%.2f" x*y
Value=6.28 format string expression

value=format fmt expression

v1=100.3

v2=200.234

title=format "%.2f Million (USD)" v1

subtitle=format "Total value: %.2f" v1+v2

ctext title 80 30 4 "sans" "maroon"

ctext subtitle 80 20 3 "sans" "gray"

100.30 Million (USD)

Total value: 300.53



value=random min max



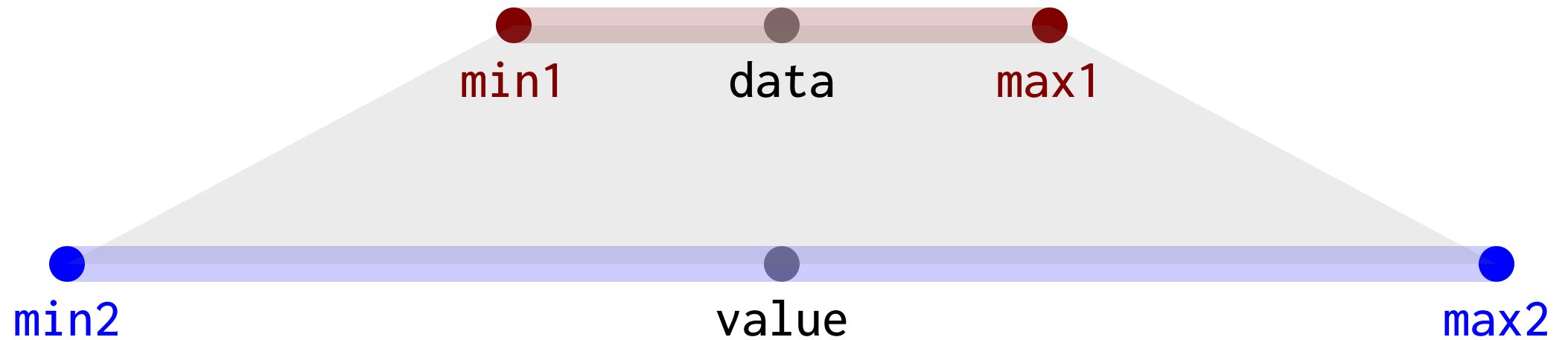
```
rx1=random 5 30  
ry1=random 15 35  
circle rx1 ry1 3 "maroon"
```



```
rx2=random 40 60  
ry2=random 15 35  
circle rx2 ry2 3 "green"
```



```
rx3=random 75 95  
ry3=random 15 35  
circle rx3 ry3 3 "blue"
```



`value=vmap data min1 max1 min2 max2`

```
yrmin=1776
yrmax=2021
smin=60
smax=90
vp=vmap 1945 yrmin yrmax smin smax
line smin 20 smax 20 0.5 "gray" 20
circle smin 20 1
circle smax 20 1
circle vp 20 2 "maroon"
```




```
data "file.d" ← data file
first 20
second 100
third 200
edata
```

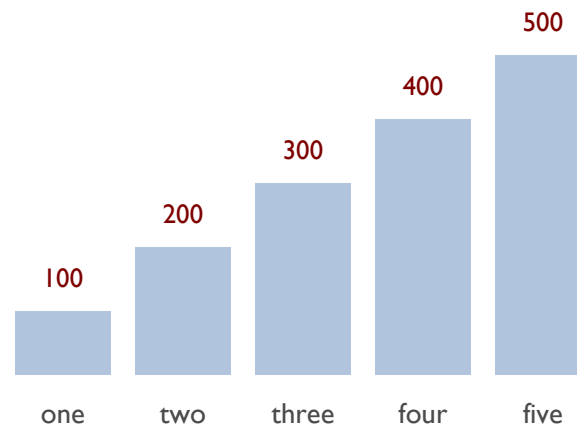
} data values

```
data "file" ... edata
```

```
data "test.d"
  one 100
  two 200
  three 300
  four 400
  five 500
```

```
edata
```

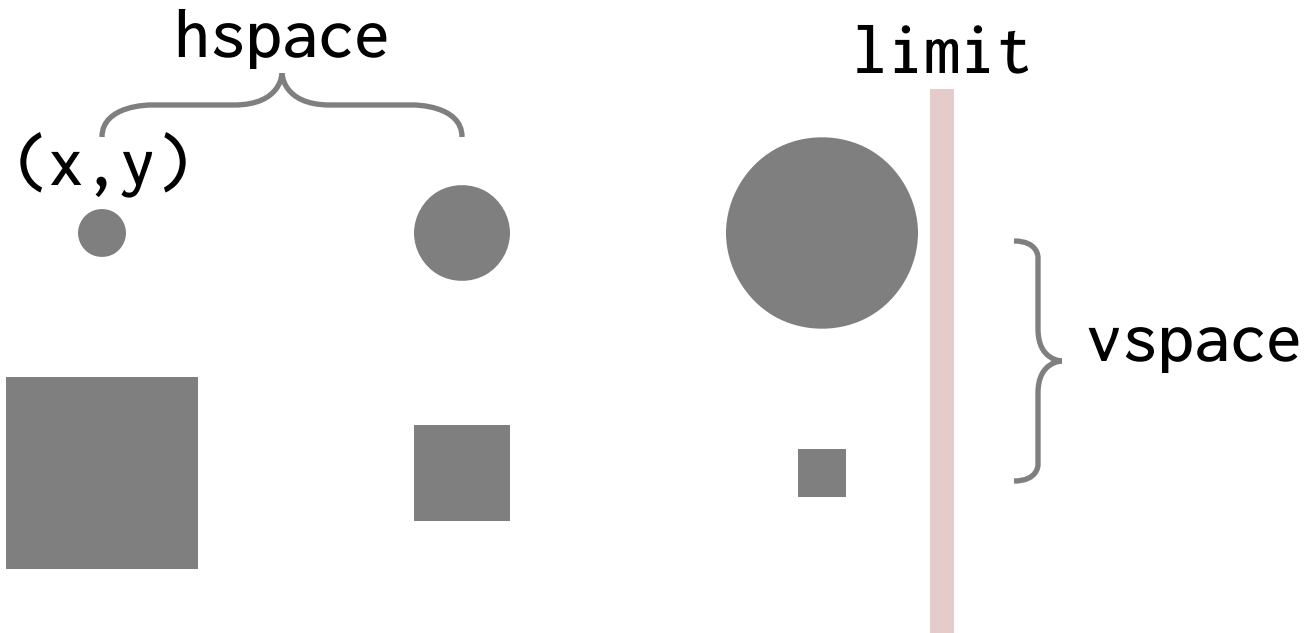
```
dchart -bar -left 50 -bottom 15 -right 70 -top 35 "test.d"
```



file

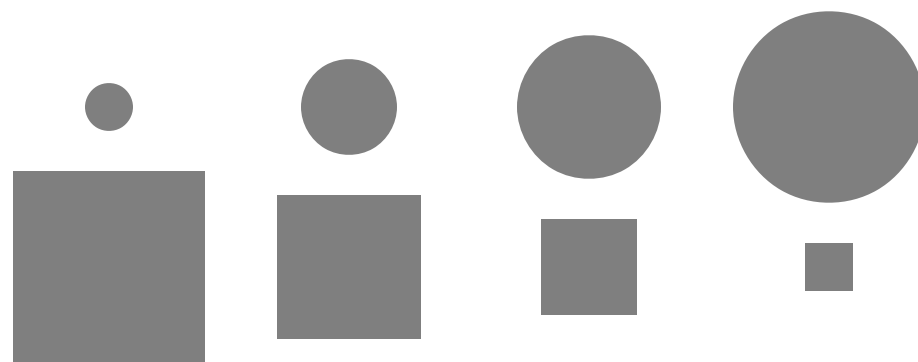


```
circle x y 2  
circle x y 4  
circle x y 8  
square x y 8  
square x y 4  
square x y 2
```



grid "file" x y hspace vspace limit

```
circle x y 2  
circle x y 4  
circle x y 6  
circle x y 8  
square x y 8  
square x y 6  
square x y 4  
square x y 2
```



grid "code/grid-ex.dsh" 35 33 10 10 65

Charts

description	keyword	arguments
Charts	dchart	options "file" (see next page)
Chart Legends	legend	"text" x y size font color

Chart Types

-bar	true	bar chart
-wbar	false	word bar chart
-hbar	false	horizontal bar chart
-donut	false	donut chart
-dot	false	dot chart
-lego	false	lego chart
-line	false	line chart
-pgrid	false	proportional grid
-pmap	false	proportional map
-bowtie	false	bowtie chart
-fan	false	fan chart
-radial	false	radial chart
-scatter	false	scatter chart
-slope	false	slope chart
-vol	false	volume (area) chart

Chart Elements

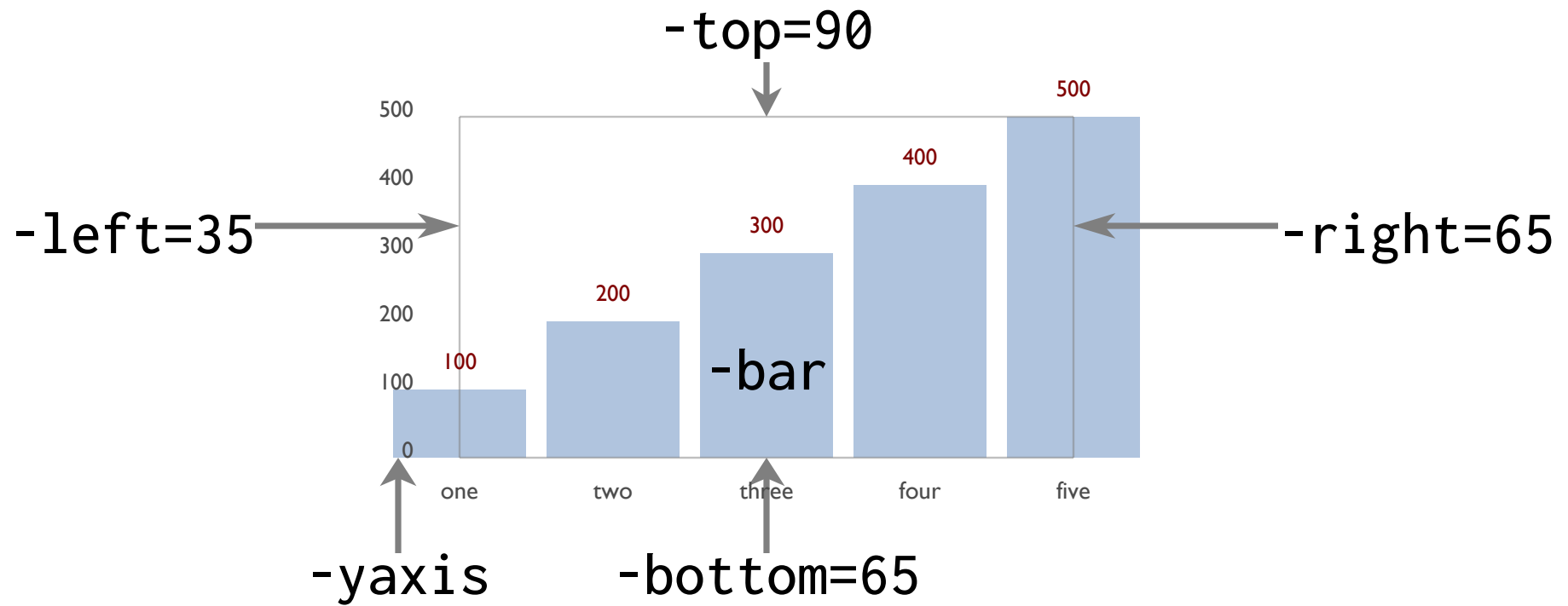
-csv	false	read CSV files
-frame	false	show a colored frame
-fulldeck	true	generate full deck markup
-grid	false	show gridlines on the y axis
-note	true	show annotations
-pct	false	show computed percentage
-rline	false	show a regression line
-solidpmap	false	show solid pmap colors
-spokes	false	show spokes in radial chart
-title	true	show the title
-val	true	show values
-xlast	false	show the last x label
-xstagger	false	stagger x axis labels
-yaxis	false	show a y axis
-chartitle	override title in data	specify the title
-datacond	low,high,color	conditional data colors
-hline	value,label	label horizontal line at value
-valpos	t=top, b=bottom, m=middle	value position
-xlabel	default=1, 0 to suppress	x axis label interval
-yrange	min,max.step	specify the y axis label range

Position and Scaling

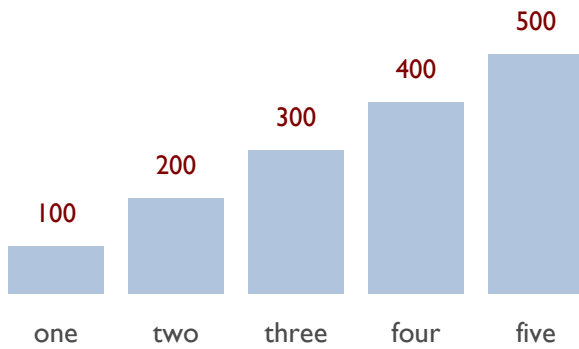
-top	80	top of the chart
-bottom	30	bottom of the chart
-left	20	left margin
-right	80	right margin
-min	data min	set the minimum data value
-max	data max	set the maximum data value

Measures and Attributes

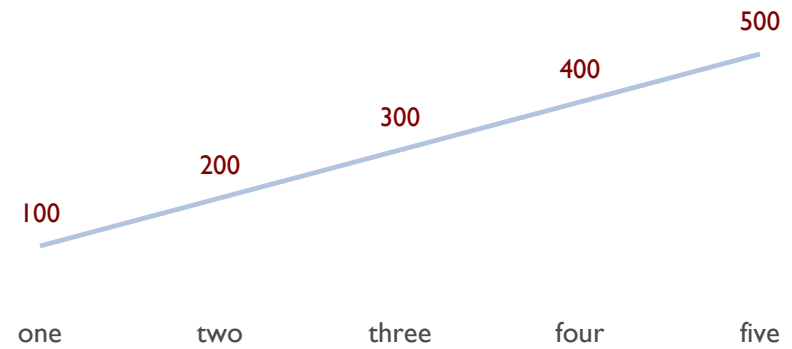
-bgcolor	white	background color
-barwidth	computed from data size	barwidth
-color	lightsteelblue	data color
-csvcol	label1,label2	specify csv columns
-datafmt	%.1f	data format for values
-dmin	false	use data minimum, not zero
-framecolor	rgb(127,127,127)	frame color
-lcolor	rgb(75,75,75)	label color
-linewidth	0.2	linewidth
-ls	2.4	linespacing
-noteloc	c=center, r=right, l=left	annotation location
-pmlen	20	pmap label length
-psize	30	diameter of the donut
-pwidth	3	width of the donut or pmap
-rlcolor	rgb(127,0,0)	regression line color
-textsize	1.5	text size
-xlabrot	0	xlabel rotation (deg.)
-vcolor	rgb(127,0,0)	value color
-volop	50	volume opacity %



dchart options "file"



```
dchart -left=10 -right=30 -top=35 -bottom=20 "test.d"
```



```
dchart -left=55 -right=85 -top=35 -bottom=20 -bar=f -line "test.d"
```



■ My text
(x, y)

legend x y fontsize font color

■ Item on the chart

■ *Thing*

```
legend "Item on the chart" 20 30 3 "sans" "red"
```

```
legend "Thing" 70 30 2 "serif" "blue"
```