## Graph Plotting

In this section we describe features for plotting graphs with tick marks and labels. Here is a simple example of how to plot a graph:

```
from visual.graph import * # import graphing features


funct1 = gcurve(color=color.cyan) # a connected curve object


for x in arange(0., 8.1, 0.1): # x goes from 0 to 8
    funct1.plot(pos=(x,5.*cos(2.*x)*exp(-0.2*x))) # plot
```

Importing from visual.graph makes available all Visual objects plus the graph plotting module. The graph is autoscaled to display all the data in the window.

A connected curve (gcurve) is just one of several kinds of graph plotting objects. Other options are disconnected dots (gdots), vertical bars (gvbars), horizontal bars (ghbars), and binned data displayed as vertical bars (ghistogram; see later discussion). When creating one of these objects, you can specify a color attribute. For gvbars, ghbars, and ghistogram you can also specify a delta attribute, which specifies the width of the bar (the default is delta=1. for gvbars and ghbars, or the width of a bin for ghistogram).

You can plot more than one thing on the same graph:

```
funct1 = gcurve(color=color.cyan)
funct2 = gvbars(delta=0.05, color=color.blue)
for x in arange(0., 8.1, 0.1):

    funct1.plot(pos=(x,5.*cos(2.*x)*exp(-0.2*x))) # curve
    funct2.plot(pos=(x,4.*cos(0.5*x)*exp(-0.1*x))) # vbars
```

In a plot operation you can specify a different color to override the original setting:

```
mydots.plot(pos=(x1,y1), color=color.green)
```

When you create a gcurve, gdots, gvbars, or ghbars object, you can provide a list of points to be plotted, just as is the case with the ordinary curve object:

```
points = [(1,2), (3,4), (-5,2), (-5,-3)]
data = gdots(pos=points, color=color.blue)
```

This list option is available only when creating the gdots object.

### Overall gdisplay options

You can establish a gdisplay to set the size, position, and title for the title bar of the graph window, specify titles for the x and y axes, and specify maximum values for each axis, before creating gcurve or other kind of graph plotting object:

```
graph1 = gdisplay(x=0, y=0, width=600, height=150,
          title='N vs. t', xtitle='t', ytitle='N',
          xmax=50., xmin=-20., ymax=5E3, ymin=-2E3,
          foreground=color.black, background=color.white)
```

In this example, the graph window will be located at (0,0), with a size of 600 by 150 pixels, and the title bar will say 'N vs. t'. The graph will have a title 't' on the horizontal axis and 'N' on the vertical axis. Instead of autoscaling the graph to display all the data, the graph will have fixed limits. The horizontal axis will extend from -20. to +50., and the vertical axis will extend from -2000. to +5000. (xmin and ymin must be negative; xmax and ymax must be positive.) The foreground color (white by default) is black, and the background color (black by default) is white. If you simply say gdisplay(), the defaults are x=0, y=0, width=800, height=400, no titles, fully autoscaled.

Every gdisplay has the attribute display, so you can manipulate basic display aspects of the graphing window:

```
graph1.display.visible = 0 # make the display invisible
```

You can have more than one graph window: just create another gdisplay. By default, any graphing objects created following a gdisplay belong to that window. You can also specify which window a new object belongs to:

```
energy = gdots(gdisplay=graph1, color=color.blue)
```

## Histograms (sorted, binned data)

The purpose of ghistogram is to sort data into bins and display the distribution. Suppose you have a list of the ages of a group of people, such as [5, 37, 12, 21, 8, 63, 52, 75, 7]. You want to sort these data into bins 20 years wide and display the numbers in each bin in the form of vertical bars. The first bin (0 to 20) contains 4 people [5, 12, 8, 7], the second bin (20 to 40) contains 2 people [21, 37], the third bin (40 to 60) contains 1 person [52], and the fourth bin (60-80) contains 2 people [63, 75]. Here is how you could make this display:

```
from visual.graph import *
.....
agelist1 = [5, 37, 12, 21, 8, 63, 52, 75, 7]
ages = ghistogram(bins=arange(0, 80, 20), color=color.red)
ages.plot(data=agelist1) # plot the age distribution
.....
ages.plot(data=agelist2) # plot a different distribution
```

You specify a list (bins) into which data will be sorted. In the example given here, bins goes from 0 to 80 by 20’s. By default, if you later say

```
ages.plot(data=agelist2)
```

the new distribution replaces the old one. If on the other hand you say

```
ages.plot(data=agelist2, accumulate=1)
```

the new data are added to the old data.

If you say the following,

```
ghistogram(bins=arange(0,50,0.1), accumulate=1, average=1)
```

each plot operation will accumulate the data and average the accumulated data. The default is no accumulation and no averaging.

## gdisplay vs. display

A gdisplay window is closely related to a display window. The main difference is that a gdisplay is essentially two-dimensional and has nonuniform x and y scale factors. When you create a gdisplay (either explicitly, or implicitly with the first gcurve or other graphing object), the current display is saved and restored, so that later creation of ordinary Visual objects such as sphere or box will correctly be associated with a previous display, not the more recent gdisplay.