Polar and Rectangular coordinate Conversions and Rotations

Definitions:

Polar coordinates are when you have a point in 2D space and from that point you know a distance and the angle. the starting point is typically from your universe's centre 0, 0.

Rectangular coordinates (Also known as the Cartesian coordinate system) are when you have the exact x and y location which is relative to the centre of your universe 0, 0.

Details
Conversion between these 2 systems can be a challenge if you don't have the correct mathematical background.

The need for this conversion could be for many reasons. For example you may be tracking the heading and direction of a moving object along a straight line, and need to calculate the x and y location for this object as it makes it's journey.

To work out the delta offsets from your X and Y location using a angle and distance.

Polar conversion example
```
double angle;
double distance
// … Code etc…
angle = 40.0; // Angle in degrees
distance = 2.0;
// …

double a;
double x_off;
double y_off;
a = (2*M_PI)/360*angle; // (2*M_PI)/360 calculates how many degree's there are to a radian
// (as sin and cos use radians not degrees.)
x_off = cos(a) * distance;
y_off = sin(a) * distance;
// Now use x_off and y_off to translate/alter whatever you are tracking.
// I.e. Add this to your object's x and y location.
```

Rectangular conversion example
Another type of information you may require is the distance to some other object in the space.
e.g. This may be for a radar, remember your object may not be travelling directly to this object.

```
// Source object
double x1=8;
double y1=9;

// Target object
double x2=45;
double y2=32;

// … code …

// The location of the target if source vector was treated as 0, 0
double x3;
double y3;
```

```
// magnitude stores the distance to the target.
double magnitude;
// target_angle contains the angle to the target.
double target_angle;
double xtemp;
double ytemp;
x3 = x2 - x1;
y3 = y2 - y1;
// Force value to always be positive.
xtemp = fabs(x3);
ytemp = fabs(y3);

magnitude = 1/sqrt(xtemp*xtemp+ytemp*ytemp);

xtemp = x3*magnitude;
ytemp = y3*magnitude;

target_angle = atan2(ytemp, xtemp) / ((2*M_PI)/360);

// consume magnitude and target_angle in some way
```

Another related use could be when you have an object described in an array of x, y pairs or even a 2 dimensional array you wish to rotate. For this you would iterate through your array and recalculate the new position for each x, y pair.
* Remember though that multiple pixels may end up in the same destination due to rounding issues.
* The rotated object will also protrude outside of the dimensions (bounding box) of your original array.
* The

Tips:
1. Never loose your normal un-rotated data, always rotate into other storage. If this is not done the loss of precision in the floating point storage will accumulate and gradually distort your object to the point where it will no longer be recognisable.
2. Maths functions such as sin, cos, sqrt and atan2 can be computationally expensive and many repeated operations can be slow for your computer to perform. Investigating you math libraries can be worth the effort. Where possible offload the heavy calculations to the hardware by utilising existing technology e.g. dedicated 3D hardware (Through OpenGL or DirectX).
3. Remember most of the maths functions work in radians not degrees.
4. You may collapse code to reduce variable use.
5. Make use of pre calculate static values, e.g. A degree in radians is never going to alter. Why make the CPU do this extra work?

References:
Vectors and Their Applications, A.J.PETTOFREZZO, PRENTICE HALL
Maths in Society Geometry, M.J.SHIELD, Jacaranda
11b Q maths, R.Brodie, MORTON BAY PUBLISHING
http://en.wikipedia.org/wiki/Polar_coordinate_system
http://en.wikipedia.org/wiki/Cartesian_coordinate_system
http://en.wikipedia.org/wiki/List_of_canonical_coordinate_transformations

Document Author
Fred Houweling
fred@houweling.com.au
http://www.houweling.com.au/

Latest document version stored at:
http://www.houweling.com.au/r/programming

27 Jan 2008