Vector 2d library

Table of Contents

Closely related to Complex numbers (a library & demonstrations) by tenochtitlanuk.
Uses ATAN2() function coded by Stefan Pendl (thread Collision detection maths error, reply 6)

There are times then you need add vectors, scale vectors, decompose vectors into normal and tangential parts.
In these times, this library might prove handy.
Of cource calling functions take time so everything works slower - but they say
"Make it RUN, make it run RIGHT, then make it run FAST".

Vectors are stored in a string as a space-separated pair of numbers.
So there is roundoff errors. Also, ther are no checks if length of vector is null - so function vectUnit$(v$) might fail on that.
As slight bonus, you can use vectors "as is" in graphic commands.
Like this

```
v1$=vect$(100,100)
v2$=vect$(300,200)
main.graphicbox1 "line ";v1$;" ";v2$
```

Or even like this

```
offset$=vect$(100,100)
v1$=vect$(1,1)
v2$=vect$(3,2)
main.graphicbox1 "line ";vectAdd$(offset$,vectScale$(10,v1$));" "
;vectAdd$(offset$,vectScale$(10,v2$))
```

Console (text mode) demo

```
'vector 2d lib demo
'vectors stored as "x y" pairs, (to be splitted by Word$)
'by tsh73, Feb 2013
```

```
'creating a vector
v1$=vect$(3,4)
print "New vector created: ";v1$
print

'copying a vector. Just assign to string variable
v2$=v1$

print "Getting a components of a vector:"
print vectX(v1$)
print vectY(v1$)
print

print "Length of a vector:"
print vectLen(v1$)
print

print "Unit vector (length=1) with same direction:"
u1$=vectUnit$(v1$)
print u1$
print "and it's length is (as should be):"
print vectLen(u1$)
print

print "Adding vectors"
print "let's make another vector"
v3$=vect$(1,-2)
print v3$
print "The sum (";v1$;") + (";v3$;") is"
print vectAdd$(v1$,v3$)
print

print "Subtracting same two vectors"
print "(";v1$;") - (";v3$;") is"
print vectSub$(v1$,v3$)
print

print "Dot product of same two vectors"
print "(";v1$;")*(";v3$;") is"
print vectDotProduct(v1$,v3$)
print "as a side note, it is 0 for perpendicular vectors"
print

print "Scaling a vector"
print "by half"
print vectScale$(0.5,v1$)
```

```
print "3x"
print vectScale$(3,v1$)
print "reverse vector by multiplying it by -1"
print vectScale$(-1,v1$)
print

print "We can decompose any vector into sum "
print "of normal and tangential parts along any direction"
print "First, let's try along OX axis"
base$ = vect$(1,0)
print "The direction is "; base$
t$=vectTangent$(v1$,base$)
print "Tangential part is ";t$
n$=vectNorm$(v1$,base$)
print "Normal part is ";n$
print "Their sum is "; vectAdd$(t$,n$)
print "(same as initial vector)"
print
print "Now try it with another direction"
base$ = v3$
print "The direction is "; base$
t$=vectTangent$(v1$,base$)
print "Tangential part is ";t$
n$=vectNorm$(v1$,base$)
print "Normal part is ";n$
print "Their sum is "; vectAdd$(t$,n$)
print "(should be same as initial vector)"
print "(Well, you see there is roundoff errors possible)"
print

print "Angle between vector and OX axis, radians"
print vectAngle(v1$)
print

print "So with length and angle, we can convert to polar coords"
print "Vector ";v1$;" is"
print "Polar radius and angle "
r=vectLen(v1$)
a=vectAngle(v1$)
print r, a
print

print "There is a function to convert from polar to cartesian"
print vectFromPolar$(r, a)
print "(should be same as initial vector)"
print "Some other vector: length 7 at angle 60 degrees"
```

```
r=7
a=60*acs(-1)/180    'acs(-1)==pi
print vectFromPolar$(r, a)
print

print "Rotating vector by arbitrary agle"
print "by 30 degrees"
print vectRotate$(v1$,30*acs(-1)/180)
print "by 90 degrees"
a=90*acs(-1)/180    'acs(-1)==pi, so it's actually pi/2
print vectRotate$(v1$,a)
print "by -90 degrees"
print vectRotate$(v1$,0-a)  'JB doesn't allow "-a"
print "by 180 degrees"
print vectRotate$(v1$,180*acs(-1)/180)
print "(Well, easier to myltiply by -1)"
print
print "*That's all, folks.*"
end
'==================================
'vector 2d lib
'vectors as "x y" pairs, to be splitted by Word$
'by tsh73, Feb 2013
function vect$(x,y)
    vect$=x;" ";y
end function

function vectX(v$)
    vectX=val(word$(v$,1))
end function

function vectY(v$)
    vectY=val(word$(v$,2))
end function

function vectLen(v$)
    x=val(word$(v$,1))
    y=val(word$(v$,2))
    vectLen=sqr(x*x+y*y)
end function

function vectUnit$(v$)
    x=val(word$(v$,1))
    y=val(word$(v$,2))
    vectLen=sqr(x*x+y*y)
    vectUnit$=x/vectLen;" ";y/vectLen
```

```
end function

function vectAdd$(v1$,v2$)
    x1=val(word$(v1$,1))
    y1=val(word$(v1$,2))
    x2=val(word$(v2$,1))
    y2=val(word$(v2$,2))
    vectAdd$=x1+x2;" ";y1+y2
end function

function vectSub$(v1$,v2$)
    x1=val(word$(v1$,1))
    y1=val(word$(v1$,2))
    x2=val(word$(v2$,1))
    y2=val(word$(v2$,2))
    vectSub$=x1-x2;" ";y1-y2
end function

function vectDotProduct(v1$,v2$)
    x1=val(word$(v1$,1))
    y1=val(word$(v1$,2))
    x2=val(word$(v2$,1))
    y2=val(word$(v2$,2))
    vectDotProduct=x1*x2+y1*y2
end function

function vectScale$(a,v$)   'a * vector v$
    x=val(word$(v$,1))
    y=val(word$(v$,2))
    vectScale$=a*x;" ";a*y
end function

function vectTangent$(v$,base$)
    n$=vectUnit$(base$)
    vectTangent$=vectScale$(vectDotProduct(n$,v$),n$)
end function

function vectNorm$(v$,base$)
    vectNorm$=vectSub$(v$,vectTangent$(v$,base$))
end function

function vectAngle(v$)
    x=val(word$(v$,1))
    y=val(word$(v$,2))
    vectAngle=atan2(y,x)
end function
```

```
function vectFromPolar$(rho, phi)
    vectFromPolar$=rho*cos(phi);" ";rho*sin(phi)
end function

function vectRotate$(v$,alpha)
    x=val(word$(v$,1))
    y=val(word$(v$,2))
    rho=sqr(x*x+y*y)
    phi=atan2(y,x)+alpha
    vectRotate$=rho*cos(phi);" ";rho*sin(phi)
end function

function dePi$(x)   'pure aestetics
    pi = acs(-1)
    dePi$=x/pi;"Pi"
end function

'----------------------------
function atan2(y,x)
    pi = acs(-1)    'could be made global to save some ticks
    if x <> 0 then arctan = atn(y/x)

    select case
        case x > 0
        atan2 = arctan

        case y>=0 and x<0
        atan2 = pi + arctan

        case y<0 and x<0
        atan2 = arctan - pi

        case y>0 and x=0
        atan2 = pi / 2

        case y<0 and x=0
        atan2 = pi / -2
    end select
end function
```

Same demo but with graphics

```
'vector 2d lib demo
```

```
' - Graphic part
'by tsh73, Feb 2013

global offset$, scale

'window and instructions
gosub [initWindow]
call waitClick

'creating a vector
v1$=vect$(3,4)
print "New vector created: ";v1$
print
'   drawing part
gosub [axes]
call drawVector v1$
call waitClick

'copying a vector. Just assign to string variable
v2$=v1$

print "Getting a components of a vector:"
print vectX(v1$)
print vectY(v1$)

#gr "color green"
call drawVector vect$(vectX(v1$), 0)
#gr "color blue"
call drawVector vect$(0, vectY(v1$))
call waitClick

print "Length of a vector:"
print vectLen(v1$)
print

print "Unit vector (length=1) with same direction:"
u1$=vectUnit$(v1$)
print u1$
print "and it's length is (as should be):"
print vectLen(u1$)
print

#gr "color cyan"
call drawVector u1$
call waitClick
```

```
print "Adding vectors"
print "let's make another vector"
v3$=vect$(1,-2)
print v3$
print "The sum (";v1$;") + (";v3$;") is"
print vectAdd$(v1$,v3$)
print

gosub [axes]
call drawVector v1$
#gr "color green"
call drawVector v3$
#gr "color blue"
call waitClick
call drawVector vectAdd$(v1$,v3$)
call waitClick

print "Subtracting same two vectors"
print "(";v1$;") - (";v3$;") is"
print vectSub$(v1$,v3$)
print

gosub [axes]
call drawVector v1$
#gr "color green"
call drawVector v3$
call waitClick
#gr "color blue"
call drawVector vectSub$(v1$,v3$)
call waitClick

print "Dot product of same two vectors"
print "(";v1$;")*(";v3$;") is"
print vectDotProduct(v1$,v3$)
print "as a side note, it is 0 for perpendicular vectors"
print

print "Scaling a vector"
print "by half"
print vectScale$(0.5,v1$)
print "3x"
print vectScale$(3,v1$)
print "reverse vector by multiplying it by -1"
print vectScale$(-1,v1$)
print
```

```
gosub [axes]
call drawVector v1$
#gr "color green"
#gr "size 3"
call drawVector vectScale$(0.5,v1$)
call waitClick
#gr "color blue"
#gr "size 1"
call drawVector vectScale$(3,v1$)
call waitClick
#gr "color cyan"
#gr "size 2"
call drawVector vectScale$(-1,v1$)
call waitClick

print "We can decompose any vector into sum "
print "of normal and tangential parts along any direction"
print "First, let's try along OX axis"
base$ = vect$(1,0)
print "The direction is "; base$
t$=vectTangent$(v1$,base$)
print "Tangential part is ";t$
n$=vectNorm$(v1$,base$)
print "Normal part is ";n$
print "Their sum is "; vectAdd$(t$,n$)
print "(same as initial vector)"
print

gosub [axes]
call drawVector v1$
#gr "size 4"
#gr "color cyan"
call drawVector base$
call waitClick
#gr "size 2"
#gr "color blue"
call drawVector t$
call waitClick
#gr "color green"
call drawVector n$
call waitClick

print "Now try it with another direction"
base$ = v3$
print "The direction is "; base$
t$=vectTangent$(v1$,base$)
```

```
print "Tangential part is ";t$
n$=vectNorm$(v1$,base$)
print "Normal part is ";n$
print "Their sum is "; vectAdd$(t$,n$)
print "(should be same as initial vector)"
print "(Well, you see there is roundoff errors possible)"
print

gosub [axes]
call drawVector v1$
#gr "size 4"
#gr "color cyan"
call drawVector base$
call waitClick
#gr "size 2"
#gr "color blue"
call drawVector t$
call waitClick
#gr "color green"
call drawVector n$
call waitClick

print "Angle between vector and OX axis, radians"
print vectAngle(v1$)
print

print "So with length and angle, we can convert to polar coords"
print "Vector ";v1$;" is"
print "Polar radius and angle "
r=vectLen(v1$)
a=vectAngle(v1$)
print r, a
print

print "There is a function to convert from polar to cartesian"
print vectFromPolar$(r, a)
print "(should be same as initial vector)"
print "Some other vector: length 7 at angle 60 degrees"
r=7
a=60*acs(-1)/180    'acs(-1)==pi
print vectFromPolar$(r, a)
print

gosub [axes]
call drawVector v1$
call waitClick
```

```
#gr "color blue"
call drawVector vectFromPolar$(r, a)
call waitClick

print "Rotating vector by arbitrary agle"
print "by 30 degrees"
print vectRotate$(v1$,30*acs(-1)/180)
print "by 90 degrees"
a=90*acs(-1)/180    'acs(-1)==pi, so it's actually pi/2
print vectRotate$(v1$,a)
print "by -90 degrees"
print vectRotate$(v1$,0-a)  'JB doesn't allow "-a"
print "by 180 degrees"
print vectRotate$(v1$,180*acs(-1)/180)
print "(Well, easier to myltiply by -1)"
print

gosub [axes]
call drawVector v1$
#gr "color green"
call drawVector vectRotate$(v1$,30*acs(-1)/180)
call waitClick
#gr "color blue"
call drawVector vectRotate$(v1$,a)
call waitClick
#gr "color cyan"
call drawVector vectRotate$(v1$,0-a)
call waitClick
#gr "color black"
call drawVector vectRotate$(v1$,180*acs(-1)/180)
call waitClick

#gr "place 70 200"
#gr "\";"*That's all, folks.*"
print "*That's all, folks.*"
wait

'----------------------------------------
'parts related to graphic part of the demo
[initWindow]
    UpperLeftX = 1
    UpperLeftY = 1
    WindowWidth = 400
    WindowHeight = 400

    open "Vector demo" for graphics_nsb_nf as #gr
```

```
    #gr "trapclose [quit]"
    #gr "home; down; posxy cx cy"
    'print cx, cy
    offset$ = vect$(cx, cy)
    scale = 20
    #gr "place 70, 120"
    #gr "\";"Please align this window"
    #gr "\";"along with mainwin (console)."
    #gr "\";"It will print stuff to mainwin, "
    #gr "\";"while drawing in this window."
    #gr "\";""
    #gr "\";"Use left mouse button click"
    #gr "\";"to advance."
return

sub waitClick
    #gr "flush"
    #gr "when leftButtonDown [cont]"
    wait
[cont]
    #gr "when leftButtonDown"
    exit sub
[quit] 'and we could close while waiting
    close #gr
    print "*program ended, you can close this window*"
    end
end sub

function fix$(v$)   '"fixes" coords of vector to use on screen:
'applies scaling and offset.
    'fix$ = vectAdd$(offset$, vectScale$(scale,v$))
    'really simple, isn't't?
    'Well, almost. "Y" should be reversed
    fix$=vectAdd$(offset$, vectScale$(scale, reverseY$(v$)))
end function

function reverseY$(v$)
    x=val(word$(v$,1))
    y=val(word$(v$,2))
    reverseY$=x;" ";0-y
end function

[axes]
'axes
    bounds = 7  'like, -7 .. 7
    #gr "cls"
```

```
    #gr "color black; size 1"
    #gr "line ";fix$(vect$(-1-bounds,0));" ";fix$(vect$(1+bounds,0))
    #gr "line ";fix$(vect$(0,-1-bounds));" ";fix$(vect$(0,1+bounds))
    for i = 0-bounds to bounds
        #gr "line ";fix$(vect$(i,-0.1));" ";fix$(vect$(i,0.1))
        #gr "line ";fix$(vect$(-0.1,i));" ";fix$(vect$(0.1,i))
    next
    #gr "size 2"
    #gr "color red" 'default first vector will be red, width 2
    #gr "flush"
return

sub drawVector v$
    #gr "line ";fix$(vect$(0,0));" ";fix$(v$)
end sub

[quit] close #gr
    print "*program ended, you can close this window*"
    end

'=================================
'vector 2d lib
'vectors as "x y" pairs, to be splitted by Word$
'by tsh73, Feb 2013
function vect$(x,y)
    vect$=x;" ";y
end function

function vectX(v$)
    vectX=val(word$(v$,1))
end function

function vectY(v$)
    vectY=val(word$(v$,2))
end function

function vectLen(v$)
    x=val(word$(v$,1))
    y=val(word$(v$,2))
    vectLen=sqr(x*x+y*y)
end function

function vectUnit$(v$)
    x=val(word$(v$,1))
    y=val(word$(v$,2))
    vectLen=sqr(x*x+y*y)
```

```
    vectUnit$=x/vectLen;" ";y/vectLen
end function

function vectAdd$(v1$,v2$)
    x1=val(word$(v1$,1))
    y1=val(word$(v1$,2))
    x2=val(word$(v2$,1))
    y2=val(word$(v2$,2))
    vectAdd$=x1+x2;" ";y1+y2
end function

function vectSub$(v1$,v2$)
    x1=val(word$(v1$,1))
    y1=val(word$(v1$,2))
    x2=val(word$(v2$,1))
    y2=val(word$(v2$,2))
    vectSub$=x1-x2;" ";y1-y2
end function

function vectDotProduct(v1$,v2$)
    x1=val(word$(v1$,1))
    y1=val(word$(v1$,2))
    x2=val(word$(v2$,1))
    y2=val(word$(v2$,2))
    vectDotProduct=x1*x2+y1*y2
end function

function vectScale$(a,v$)   'a * vector v$
    x=val(word$(v$,1))
    y=val(word$(v$,2))
    vectScale$=a*x;" ";a*y
end function

function vectTangent$(v$,base$)
    n$=vectUnit$(base$)
    vectTangent$=vectScale$(vectDotProduct(n$,v$),n$)
end function

function vectNorm$(v$,base$)
    vectNorm$=vectSub$(v$,vectTangent$(v$,base$))
end function

function vectAngle(v$)
    x=val(word$(v$,1))
    y=val(word$(v$,2))
    vectAngle=atan2(y,x)
```

```
end function

function vectFromPolar$(rho, phi)
    vectFromPolar$=rho*cos(phi);" ";rho*sin(phi)
end function

function vectRotate$(v$,alpha)
    x=val(word$(v$,1))
    y=val(word$(v$,2))
    rho=sqr(x*x+y*y)
    phi=atan2(y,x)+alpha
    vectRotate$=rho*cos(phi);" ";rho*sin(phi)
end function

function dePi$(x)   'pure aestetics
    pi = acs(-1)
    dePi$=x/pi;"Pi"
end function

'----------------------------
function atan2(y,x)
    pi = acs(-1)    'could be made global to save some ticks
    if x <> 0 then arctan = atn(y/x)

    select case
        case x > 0
        atan2 = arctan

        case y>=0 and x<0
        atan2 = pi + arctan

        case y<0 and x<0
        atan2 = arctan - pi

        case y>0 and x=0
        atan2 = pi / 2

        case y<0 and x=0
        atan2 = pi / -2
    end select
end function
```