

Forum:[Hack Forums](#)

Thread: [Point \(Vector\) Movment Computing](#)

Das Kartoffel - January 9, 2017, 5:30 am

Hey guys,

I got to compute the position of a moving point in a map going from (0;0) to (100;100).
After looking online, I found $X = \cos$ of the angle (in radian) and $Y = \sin$ of the angle (in radian).

Here is what I've been doing with base position being (2;2):

PHP Code

```
<?php
float pos_move(2DVector pos, float angle)
{
    2DVector new_pos;
    float a_rad;

    new_pos = pos;
    a_rad = (angle * (M_PI / 180));
    new_pos.x += cos(angle) * 1;
    new_pos.y += sin(angle) * 1;
    printf(
"angle (degree):\t%f\nangle (radian):\t%f\ncos(%f):\t%f\nsin(%f):\t%f\nNew Pos:\t[%d][%d]\n\n",
angle, a_rad, a_rad, cos(a_rad), a_rad, sin(a_rad), (int)new_pos.x, (int)new_pos.y);
    return 0;
}
```

Here are the results with four different angles (East, North, West & South):

```
angle (degree): 0.000000
angle (radian): 0.000000
cos(0.000000): 1.000000
sin(0.000000): 0.000000
New Pos: [3][2] /* Good /
```

```
angle (degree): 90.000000
angle (radian): 1.570796
cos(1.570796): -0.000000
sin(1.570796): 1.000000
New Pos: [1][2] / Not Good: Should be [2][1] /
```

```
angle (degree): 180.000000
angle (radian): 3.141593
cos(3.141593): -1.000000
sin(3.141593): -0.000000
New Pos: [1][1] / Not Good: Should be [1][2] /
```

```
angle (degree): 270.000000
```

angle (radian): 4.712389
cos(4.712389): 0.000000
sin(4.712389): -1.000000
New Pos: [2][1] / Not Good: Should be [2][3] /*

I've also tried inverting sin & cos in the code, but also had errors.
I'm really not into maths, there must be something I don't get.
Do you know what's wrong with my code ?

Thanks.

Kantos - January 9, 2017, 6:30 am

The formula for a line can be written in this form: $y = mx + b$

where y is the y coordinate, m is the slope of the line, x is the x coordinate, and b is the y intercept.

In your case since you're just starting with a point and an angle, $m = \tan(\text{angle})$ and $b = \text{start_y} - m * \text{start_x}$ so now you have a complete formula for the line.

Das Kartoffel - January 9, 2017, 6:44 am

I'm not sure I get it as (I'm a real rubbish when it comes to maths).

In my case I don't have to move a line, but a point, which got to move the way the angle is directing it to.

Isn't:

```
new_pos.x += cos(angle);  
new_pos.y += sin(angle);
```

alright ?

Annabelle Lee - January 9, 2017, 7:07 am

(Jan 9, 2017 07:44 AM)Das Kartoffel Wrote: [\[CITE\]](#) I'm not sure I get it as (I'm a real rubbish when it comes to maths).

In my case I don't have to move a line, but a point, which got to move the way the angle is directing it to.

Isn't:

```
new_pos.x += cos(angle);  
new_pos.y += sin(angle);
```

alright ?

If you can clarify slightly I can help. Question is: You start at point (x,y) correct? And are given an angle for direction it has to move in. Are the new points always integers (whole numbers) and do you always move one unit away, or can you move more than one unit?

Das Kartoffel - January 9, 2017, 7:14 am

(Jan 9, 2017 08:07 AM)AnnabelleLee Wrote: [\[CITE\]](#) If you can clarify slightly I can help. Question is: You start at point (x,y) correct? And are given an angle for direction it has to move in. Are the new points always integers (whole numbers) and do you always move one unit away, or can you move more than one unit?

That's it, I start at a coordinate (in my example, I've set it to (2;2)).

There is only one point, which is the one we are moving, concerning the speed we actually don't mind.

Here, the problem is what I did is wrong as my point goes in bad directions.

For 90° you can see it changes X while it should change Y (therefore going up instead of going left).

Annabelle Lee - January 9, 2017, 7:17 am

(Jan 9, 2017 08:14 AM)Das Kartoffel Wrote: [\[CITE\]](#) That's it, I start at a coordinate (in my example, I've set it to (2;2)).

There is only one point, which is the one we are moving, concerning the speed we actually don't mind.

Here, the problem is what I did is wrong as my point goes in bad directions.

For 90° you can see it changes X while it should change Y (therefore going up instead of going left).

I meant, like you start at (2,2). When you change point is it possible for it to be: (2.5,2) <- decimals? (unlikely I think.) Is it limited to changing only to say (2,1) or (3,2). Or can it do something like (4,2)? < skip the 3/more than 1 spot away.

Das Kartoffel - January 9, 2017, 7:47 am

(Jan 9, 2017 08:17 AM)AnnabelleLee Wrote: [\[CITE\]](#) I meant, like you start at (2,2). When you change point is it possible for it to be: (2.5,2) <- decimals? (unlikely I think.) Is it limited to changing only to say (2,1) or (3,2). Or can it do something like (4,2)? < skip the 3/more than 1 spot away.

Yes, it can be float, but it isn't relevant. As the problem currently is the directions it's going to.

To simplify the results in the tests, I made it whole.

Annabelle Lee - January 9, 2017, 9:29 am

(Jan 9, 2017 08:47 AM)Das Kartoffel Wrote: [\[CITE\]](#) Yes, it can be float, but it isn't relevant. As the problem currently is the directions it's going to.

To simplify the results in the tests, I made it whole.

Ah I believe I found the problem. There were two: One was your typecasting to int in the printf function. This caused display/rounding errors. The other was the fact that default for sin and cosine functions are in radians, not degrees in C++.

I used the formula: $4 * \text{atan}(1)$ as approximation of PI (since I don't have the constant defined, but the result is same). Here is a corrected version (Modified function to work on my PC, you'll have to redo some variables.) Also FYI your assumptions on the way the new coordinates would be are incorrect, might want to check a coordinate grid and figure out the angles. Cheers and ask if you have more questions.

```
Code #include <iostream>
```

```
#include <cmath>
```

```
using namespace std;
```

```
void pos_move(float angle){
```

```
    float a_rad = 0, x, y;
```

```
    x = y = 2;
```

```
    a_rad = (angle * ((4*atan(1)) / 180));
```

```
    x += cos(a_rad);
```

```
    y += sin(a_rad);
```

```
    printf("angle (degree):\t%f\nangle (radian):\t%f\ncos(%f):\t%f\nsin(%f):\t%f\nNew
```

```
Pos:\t[t[%f][%f]\n\n", angle, a_rad, a_rad, cos(a_rad), a_rad, sin(a_rad), x, y);
```

```
}
```

```

int main() {
    pos_move(0);
    pos_move(90);
    pos_move(180);
    pos_move(270);
    getchar();
    return 0;
}

```

Kantos - January 9, 2017, 10:27 am

(Jan 9, 2017 07:44 AM)Das Kartoffel Wrote: [\[CITE\]](#) I'm not sure I get it as (I'm a real rubbish when it comes to maths).

In my case I don't have to move a line, but a point, which got to move the way the angle is directing it to.

Isn't:

```

new_pos.x += cos(angle);
new_pos.y += sin(angle);

```

alright ?

The line represents the path that point will take if it heads at that angle.

For example, the formula for the line from (0,0) to (100,100) is simply $y = x$. The Y coordinate is the X coordinate.

The formula for a line from (0,0) to (50,150) is $y = 3x$. The Y coordinate is the X coordinate multiplied by 3.

If I want to know further locations, I just plug in a Y or X value and solve for the other value. If I want to know where the point is halfway between (0,0) and (50,150) then I do $y = 2x = 3 (50 / 2) = 50$ so (25, 75) is the halfway point.

If you want to move the point along the line then you just increment or decrement the X position and then solve for the Y position.

For example:

```

//initial position
x_coor = 0
y_coor = 0

```

```

m = tan(angle) //calculate our slope, only needs to be done once per angle

```

```

x_coor += step //increment x value by some step amount
y_coor += m * x_coor //calculate new y coordinate, you'll need to also add the y intercept if your line
doesn't go through the origin

```

aeiro - January 13, 2017, 5:48 pm

I just realized I need to go back to school to re-learn math...