



Convert Latitude and Longitude to point in 3D space

CAREERS 2.0
by stackoverflow



+



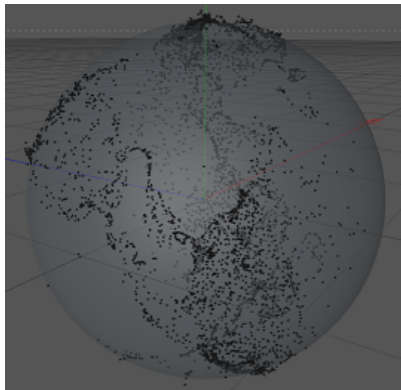
Have projects on BitBucket?
Import them easily to your profile

I need to convert latitude and longitude values to a point in the 3-dimensional space. I've been trying this for about 2 hours now, but I do not get the correct results.

The **Equirectangular** coordinates come from openflights.org. I've tried several combinations of \cos and \sin , but the result did never look like our little beloved earth.

In the following, you can see the result of applying the conversion [Wikipedia](http://en.cppreference.com/w/cpp/vector) suggests. I think one can guess from context what `c4d.Vector` is.

```
def llarToWorld(latit, longit, altid, rad):  
    x = math.sin(longit) * math.cos(latit)  
    z = math.sin(longit) * math.sin(latit)  
    y = math.cos(longit)  
    v = c4d.Vector(x, y, z)  
    v = v * altid + v * rad  
    return v
```

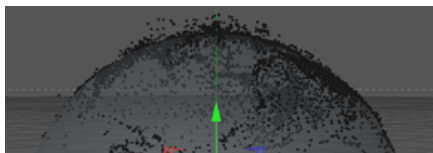


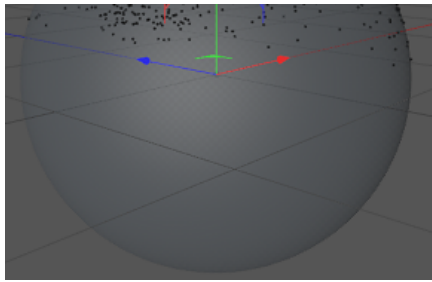
Red: X, Green: Y, Blue: Z

One can indeed identify North- and South America, especially the land around the Gulf of Mexico. However, it looks somewhat squished and kind of in the wrong place..

As the result looks somewhat rotated, I think, I tried swapping latitude and longitude. But that result is somewhat awkward.

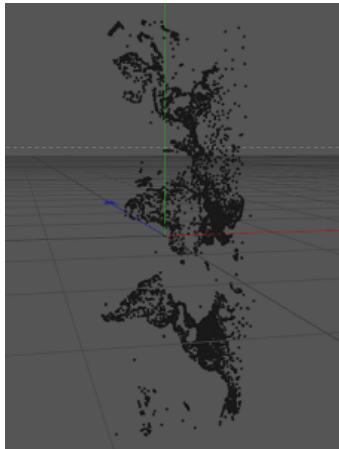
```
def llarToWorld(latit, longit, altid, rad):  
    temp = latit  
    latit = longit  
    longit = temp  
    x = math.sin(longit) * math.cos(latit)  
    z = math.sin(longit) * math.sin(latit)  
    y = math.cos(longit)  
    v = c4d.Vector(x, y, z)  
    v = v * altid + v * rad  
    return v
```





This is what the result looks like without converting the values.

```
def llarToWorld(latit, longit, altid, rad):
    return c4d.Vector(math.degrees(latit), math.degrees(longit), altid)
```



Question: How can I convert the longitude and latitude correctly?

Solution

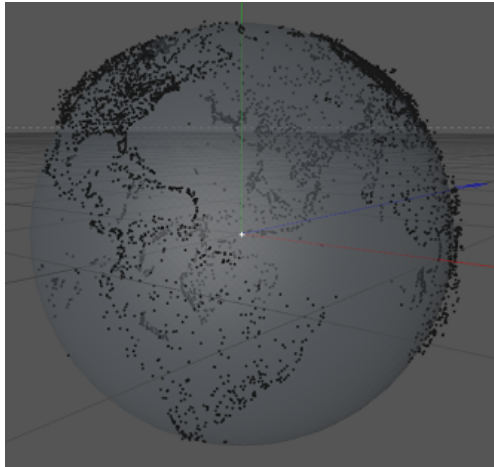
Thanks to TreyA, I found [this](http://www.mathworks.de/help/toolbox/aeroblks/llatoecefposition.html) page on mathworks.com. The code that does it's work is the following:

```
def llarToWorld(lat, lon, alt, rad):
    # see: http://www.mathworks.de/help/toolbox/aeroblks/llatoecefposition.html
    f = 0 # flattening
    ls = atan((1 - f)**2 * tan(lat)) # lambda

    x = rad * cos(ls) * cos(lon) + alt * cos(lat) * cos(lon)
    y = rad * cos(ls) * sin(lon) + alt * cos(lat) * sin(lon)
    z = rad * sin(ls) + alt * sin(lat)

    return c4d.Vector(x, y, z)
```


Actually, I switched `y` and `z` because the earth was rotated then, however, it works! That's the result:



python math 3d latitude-longitude data-conversion

edited May 9 '12 at 13:30

asked May 6 '12 at 20:07

 **Niklas R**
2,037 ●1●21●54

`altid` is the altitude, but what is `rad`? Is that the radius of earth? Are `altid` and `rad` in the same units (feet)? What if you only use the radius (i.e. just `v = v * rad`)? – [Omri Barel](#) May 6 '12 at 20:58

also look at googling 'lla to ecef' - latitude/longitude/altitude to earth-centered earth-fixed. – [TreyA](#) May 7 '12 at 11:59

- 1 @TreyA Perfect, thank you! Found this: mathworks.de/help/toolbox/aeroblks/llatoecefposition.html It is the correct formula. :) You can make your comment an answer if you want the rep, and this way I can also mark my question as answered. – [Niklas R](#) May 9 '12 at 13:25

add comment

3 Answers

you're not doing what wikipedia suggests. read it again carefully.

they say:

```
x = r cos(phi) sin(theta)
y = r sin(phi) sin(theta)
z = r cos(theta)
```

and then:

```
theta == latitude
phi == longitude
```

and, in your case, $r = \text{radius} + \text{altitude}$

so you should be using:

```
r = radius + altitude
x = r cos(long) sin(lat)
y = r sin(long) sin(lat)
z = r cos(lat)
```

note that the final entry is `cos(lat)` (you are using longitude).

answered May 6 '12 at 23:48

 **andrew cooke**
20.1k ●2●31●68

Unfortunately that is the same result as picture #2 shows.. :(– [Niklas R](#) May 9 '12 at 13:25

no it's not. lat/long are swapped only for z. you swapped them all. – [andrew cooke](#) May 9 '12 at 13:59

Well, it's not exactly the same, but equals very much the second picture. And it's very much equal to the first picture when swapping lat and long. The link I found by searching `lla to ecef` (Thanks to TreyA) is perfect. See: mathworks.de/help/toolbox/aeroblks/llatoecefposition.html – Niklas R May 9 '12 at 14:33

[add comment](#)

ANTICIPÉSE.
SUPERVISE Y GESTIONE LAS
TAREAS DE TI DE FORMA REMOTA.



I've reformatted the code that was previously mentioned here, but more importantly you have left out some of the equations mentioned in the link provided by **Niklas R**

```
def LLHtoECEF(lat, lon, alt):
    # see http://www.mathworks.de/help/toolbox/aeroblks/llatoecefposition.html

    rad = np.float64(6378137.0)      # Radius of the Earth (in meters)
    f = np.float64(1.0/298.257223563) # Flattening factor WGS84 Model
    cosLat = np.cos(lat)
    sinLat = np.sin(lat)
    FF      = (1.0-f)**2
    C       = 1/np.sqrt(cosLat**2 + FF * sinLat**2)
    S       = C * FF

    x = (rad * C + alt)*cosLat * np.cos(lon)
    y = (rad * C + alt)*cosLat * np.sin(lon)
    z = (rad * S + alt)*sinLat

    return (x, y, z)
```

Comparison output: finding ECEF for Los Angeles, CA (34.0522, -118.40806, 0 elevation)

My code:

X = -2516715.36114 meters or **-2516.715 km**
Y = -4653003.08089 meters or **-4653.003 km**
Z = 3551245.35929 meters or **3551.245 km**

Your Code:

X = -2514072.72181 meters or **-2514.072 km**
Y = -4648117.26458 meters or **-4648.117 km**
Z = 3571424.90261 meters or **3571.424 km**

Although in your earth rotation environment your function will produce *right geographic region* for display, it will **NOT** give the right ECEF equivalent coordinates. As you can see some of the parameters vary by as much as **20 KM** which is rather a large error.

Flattening factor, `f` depends on the model you assume for your conversion. Typical, model is **WGS 84**; however, there are other models.

Personally, I like to use [this link](#) to Naval Postgraduate School for sanity checks on my conversions.

answered Dec 3 '13 at 19:34



This works but you first need to convert angles to radians. And lat/long values are usually supplied as degrees. – Tomislav Muic Apr 5 at 14:41

[add comment](#)

As TreyA statet, `LLA to ECEF` is the solution. See <http://www.mathworks.de/help/toolbox/aeroblks/llatoecefposition.html>

answered May 28 '12 at 17:13



[add comment](#)

Not the answer you're looking for? Browse other questions tagged [python](#) [math](#)

[3d](#) [latitude-longitude](#) [data-conversion](#) or [ask your own question](#).

