C2 星图

2020 数据结构与算法　付皓竹、吴熙楠小组　　指导老师：陈斌

【摘要】：星图，是恒星观测的一种形象记录，它是天文学上用来认星和指示位置的一种重要工具。星图将天体的球面视位置投影于平面而绘成的图来，表示它们的位置、亮度和形态。它是天文观测的基本工具之一。有的星图只绘出恒星，有的星图则绘出各种天体。按使用对象分，有的供天文工作者使用，有的供天文爱好者使用。我们小组此次采用掌控板来模拟星图软件的部分功能。

一、选题及创意介绍

众所周知，星图，是恒星观测的一种形象记录，它是天文学上百用来认星和指示位置的一种重要工具。星图将天体的球面视位置投影于平面而绘成的图，表示它们的位置、亮度和形态。它是天文度观测的基本工具之一。星图种类繁多，有的用来辨认星星，有的用来证认某天体(或天象)，有的用来对比发生的变异等等。有的星图只绘出恒星，有的星图则绘出各种天体。按使用对象分，有的供天文版工作者使用，有的供天文爱好者使用。近世出版权的星图按出版形式分为图册和挂图。星图的方位是：上北下南，左东右西。

星图这款软件可以帮助我们学习到很多的天文方面的知识，培养对于天文的兴趣，因此，我们小组基于星图的这方面的用途，打算采用掌控板来模拟市面上的星图软件的部分功能，这个作品的一大特点便是可以利用掌控板内置的磁力计和加速度计，通过一系列的数学推导得出掌控板所指向的方位角以及起伏的俯仰角，并且能够在得到指向星星之后显示出对星星的介绍，对于天文小白也是十分友好的。

二、设计方案和硬件连接

事先输入某一时间星星的地平坐标和赤纬，用球面三角公式转化为时角坐标。再利用掌控版链接 wifi 的功能同步时间，将时间差转换为时角差，得到现在的时角坐标，再利用公式转化为地平坐标。利用掌控版的磁力计功能得到方向角和俯仰角，模拟人的观测。

三、实现方案和代码分析

A.实现方案

1、先录入某个时间的星星的位置
2、使用球面三角公式得到当前时间星星的位置
3、用掌控版的 magnetic 相关功能函数，得到地磁的 xyz 三个坐标的分量，确定掌控版的方位角和俯仰角（事先确定当地的纬度）
4、搜寻掌控版指向的那片天空的星星，并表在显示屏上
5、可以点一下搜寻一次，最好能实现每 0.1 秒刷新一次，这样星星在显示屏上的轨迹就是连续的。

B.代码分析

1.方向角的确定

```
gz0 = -9.7
while True:
    X = magnetic.get_x()
    Y = magnetic.get_y()
    Z = magnetic.get_y()
    gx = accelerometer.get_x()
    gy = accelerometer.get_y()
    gz = accelerometer.get_z()
    if gz==0 and gx==gz0:
        return h=-(math.pi)/2
    elif gz==0 and gx==-gz0:
        return h=(math.pi)/2
    else:
        return h = math.atan(-gx/gz)   # 俯仰角
    if gy==0 and gx!=0:
        if gx*math.sin(h)+gz*math.cos(h)==gz0:
            return c=(math.pi)/2
        else:
```

```
            return c=-(math.pi)/2
    elif gy==0 and gx==0:
        h=0
        return c=0
    else:
        return c = math.atan(-gx/gy)  # 侧倾角
    Xh = X*math.cos(c) - Y*math.sin(c)
    Yh = Y*math.cos(c)*math.cos(h) + Z*math.sin(h) - \
         X*math.cos(h)*math.sin(c)
    A = math.atan(Yh/Xh)*180/math.pi
    if Xh < 0:
        A = 180 - A
    elif Xh > 0 and Yh < 0:
        A = -A
    elif Xh > 0 and Yh > 0:
        A = 360 - A
    elif Xh == 0 and Yh < 0:
```

```
        A = 90
    elif Xh == 0 and Yh > 0:
        A = 270
    oled.fill(0)
    oled.DispChar(str(A), 0, 0)
    oled.DispChar(str(h*180/math.pi), 0, 16)
    oled.show()
```

我们首先利用掌控板内置的磁力计输出xyz方向上的磁感应强度大小以及重力加速度的大小，然后可以通过坐标变换计算出还原到地平坐标轴上的$x'y'z'$的换算关系，再通过对于重力加速度的计算可以得到侧倾角以及俯仰角的大小关系，最后带入磁感应强度大小的公式即可以计算出$x'y'z'$方向上的大小，最后利用方位角公式$\varphi = arctan\frac{B_y}{B_x}$即可以得到方位角。

注意：

⑴由于在公式中出现分数，因此有必要对于分母为零的情况进行讨论以免出现在某个角度上运行结果报错的情况。

⑵由于 python 计算角度输出为弧度制，因此必须将其转换为角度值进行计算。

⑶因为方向角规定范围为 0-360 °，而输出结果可能会出现负数或者其他，因此最后我们加上对于方向角分别处于四个象限内不同的讨论。

⑷在掌控板内部可能会出现有内部电流的影响，导致磁力计零点与坐标轴零点不重合的情况，因此应该有一个校零系统，具体方案如下：将掌控板水平放置，缓慢旋转 360 °，记录此过程中 xy 方向上的磁感应强度读数，取最大值和最小值平均值为 X_0 和 Y_0，再绕水平轴旋转，对 z 轴进行同样的操作，记录最大值与最小值的平均值为 Z_0，然后后续计算只需要减去校正值即可。但由于经测验，校正值相对于主值很小可以忽略，并且校正过程繁琐难以做到缓慢旋转，影响使用体验，因此我们决定将此功能删去。

2.坐标转换函数

```
Lat = 30.67 # 当地纬度

#函数一：地平坐标转化为时角坐标

def c_Dec(x,y):    # 赤纬declination，x为方位角，y为地平高度
    return float("%.4f"%(90 - arccos(cos(90 - Lat)*cos(90 - y) + sin(90 - Lat)*sin(90 - y)*cos(360 - x))))

def c_HA(x,y,z):    # 时角hour angle，x为方位角，y为地平高度，z为赤纬
    Dec2 = c_Dec(x,y + 0.0001)
    HA = float("%.4f"%(arcsin(sin(x)*sin(90 - y)/sin(90 - z))))*24/360
    HA1 = float("%.4f"%(arcsin(sin(x)*sin(90 - y - 0.0001)/sin(90 - Dec2))))*24/360
    if y == 90:
        HA = 0
    elif y == -90:
        HA = 12   # 在h为正负90时，HA与A无关，直接输出好了
    else:
        if 0 <= x < 180 and HA > HA1:
            HA = 24 - HA
        elif 0 <= x < 180 and HA <= HA1:
            HA = 12 + HA
        elif 180 <= x < 360 and HA > HA1:
            HA = 12 + HA
        elif 180 <= x < 360 and HA <= HA1:
            if HA < 0:
                HA = -HA
    return HA
```

```
#函数四：时角坐标转化为地平坐标

def c_h(x,y):   # x为赤纬，y为时角
    a = y * 15
    return float("%.4f"%(90 - arccos(cos(90 - Lat)*cos(90 - x) + sin(90 - Lat)*sin(90 - x)*cos(a))))

def c_A(x,y,z):   # x为赤纬，y为时角，z为地平高度
    a = y * 15
    h2 = c_h(x + 0.0001,y)
    A1 = float("%.4f"%(arcsin(sin(a)*sin(90 - x)/sin(90 - z))))
    A2 = float("%.4f"%(arcsin(sin(a)*sin(90 - x - 0.0001)/sin(90 - h2))))
    if 0 <= y < 12 and A1 <= A2:
        return 180 + A1
    elif 0 <= y < 12 and A1 > A2:
        return 360 - A1
    elif 12 <= y < 24 and A1 <= A2:
        if A1 < 0:
            A1 = -A1
        return A1
    elif 12 <= y < 24 and A1 > A2:
        return 180 + A1
```

在 c_HA 和 c_A 中，采取了轻微扰动的方法解决了象限问题。

但计算速度不够高，目前收录但 68 颗星星全部计算完毕将近 5 秒，于是采用了先将所有星星算一遍，再根据掌控版方位角、俯仰角输出到屏幕上。

3.星星的显示问题

```
def starShowing(target_h,target_A): # 显示星星
    global target_list
    target_list = []
    for i in list_now:
        A = i[1]
        h = i[2]
        # 搜索范围：距离单片机背面指向坐标的角距小于60度
        if distance(A,h,target_A,target_h) <= 60:
            # 求以指向坐标为原点的直角坐标系的x、y
            x = distance(A,target_h,target_A,target_h)
            y = distance(target_A,h,target_A,target_h)
            # 显示范围：-24<x<24,-12<y<12
            if x<24 and y<12:
                if (A < target_A and target_A - A < 24) or (A - target_A > 24):
                    x = - x
                d1 = distance(target_A,h,target_A,0)
                d2 = distance(target_A,target_h,target_A,0)
                if (d1 < d2 and h > 0 and target_h > 0) or (d1 > d2 and h < 0 and target_h < 0) \
                   or (h < 0 and target_h >= 0):
                    y = - y
                # 再变换到以左上角为坐标原点，便于显示.比例：24（角距）：64（像素）
                x = (x + 24)/24*64
                y = - (y - 12)/24*64
                target_list += [[i[0],x,y,dict_magnitude[i[0]]]]  # [[name, x, y, 视亮度]]
    oled.fill(0)
    for i in target_list:
        x = i[1]
        y = i[2]
        m = i[3] # 视亮度
        if m > 0.5:
            oled.pixel(round(x),round(y),1)
        else:
            oled.fill_circle(round(x),round(y),1,1)
    oled.show()
```

采用角距作为星星间的距离，以角距的水平投影和垂直投影作为 x、y 坐标。但是实际中发现，受计算时间的限制，导致不能每 0.1 秒刷新一次，于是只能采用按一下按钮刷新一次的办法。只有当掌控版的方位发声变化时才进行刷新，这样也避免了代码的重复执行。

最后，根据星星的亮度，设置在屏幕上绘制点的大小。

4.中断函数的探索

```
button_a.irq(trigger=Pin.IRQ_FALLING, handler=ledon)   #设置按键 A 中断
```

上面这条代码写在循环语句前，当触发相应时间时，会先处理中断函数 ledon。这让我们有办法在显示的过程中手动调整方向角和俯仰角、显示星星名字。

注意：在中断函数中不能再设置中断并写一个循环结构了，否则中断函数中的中断会失效，且陷入死循环。

四、后续工作展望

我们此次的掌控板制作的星图实现了初步功能：用掌控版指向哪个方向，就显示那个方向的天区的星星有哪些，标出相对位置。对于后续工作，由于在开始阶段需要将掌控板水平放置，若不是水平放置则会出现一定误差，导致方向角计算出错，因此我们可以再开始时加入一个水平仪代码，利用掌控板内置的加速度计实现，在水平时亮绿灯，不水平时亮红灯，

具体代码如下：

```
from mpython import *
x0=63
y0=31
while True:
    x=accelerometer.get_x()
    y=accelerometer.get_y()
    if y<=1 and y>=-1:
        deltaX=int(numberMap(y,1,-1,-64,64))
    if x<=1 and x>=-1:
        deltaY=int(numberMap(x,1,-1,32,-32))
    if deltaX==0 and deltaY==0:
        rgb.fill((0,10,0))
        rgb.write()
    else:
        rgb.fill((10,0,0))
        rgb.write()
    oled.show()
    oled.fill(0)
```

同时，我们还可以通过 mpython 进行音频操作，使得在指向不同位置显示出不同的星图过后播放出更加怡人的音乐，进而增强使用者的使用体验。甚至还可以通过按键实现对于当前区域星图部分的放大与缩小（具体实现方法暂时还不知道），这样能够进一步帮助使用者了解其喜欢星座的相关信息。

我们还会进一步增加星星数量，加快显示速度。更具体可行的是改进显示过程中的某些循环结构，减少重复计算；优化数据的储存，比如说星星的视亮度，不用把每颗星的数值都储存，只用分个类即可。

五、小组分工合作

付皓竹：负责数据录入以及球面坐标转换、星星显示的实现
吴熙楠：负责方位角俯仰角的计算以及片段动画制作