Soaring Over Terrain

Danielle Bolthausen,
Animesh Chaudhry,
Trenton Jansen,
Robert Rivas,
San Tran

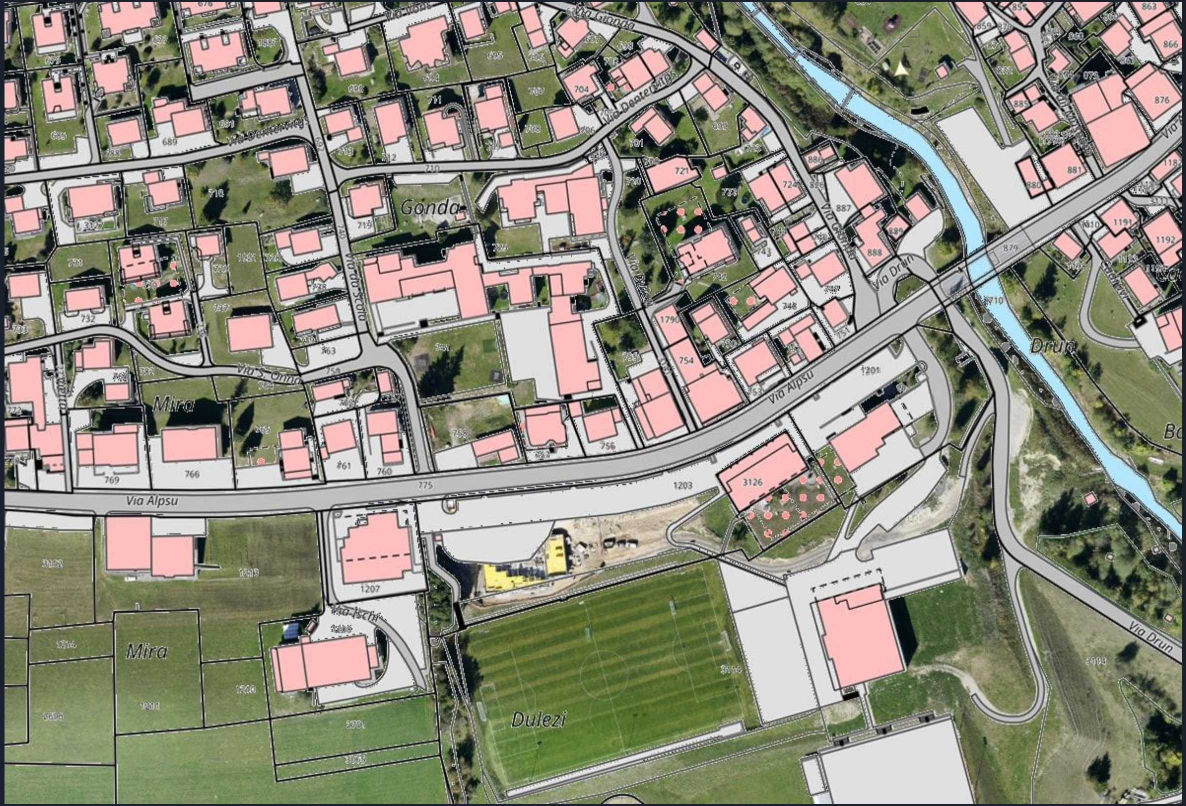
December 11, 2019
Math 370
California State University, Fullerton

Introduction

- Soaring over a terrain with a drone/camera.
- Implementation of a “camera” in Matlab
- Drone Flightpath
- Lighting effects on terrain over passage of time
- How these affect each other

Surveying & GIS

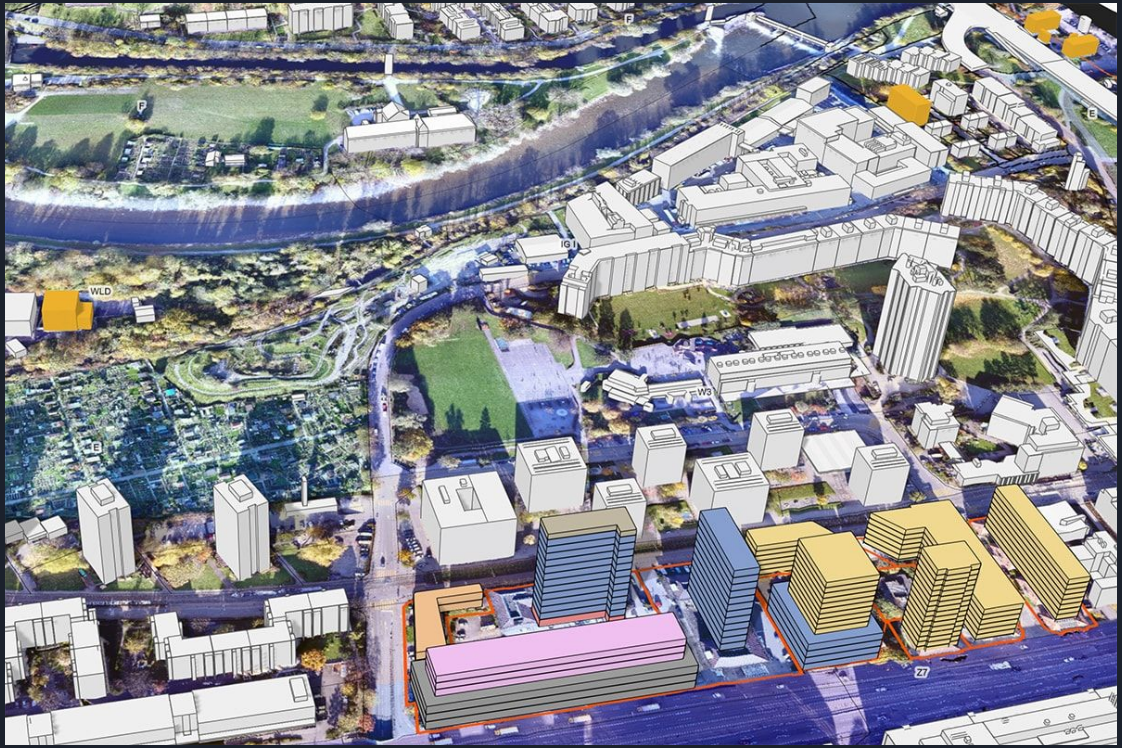
Land surveying / cartography



Land management and development

Precise measurements

Urban planning



Model Breakdown

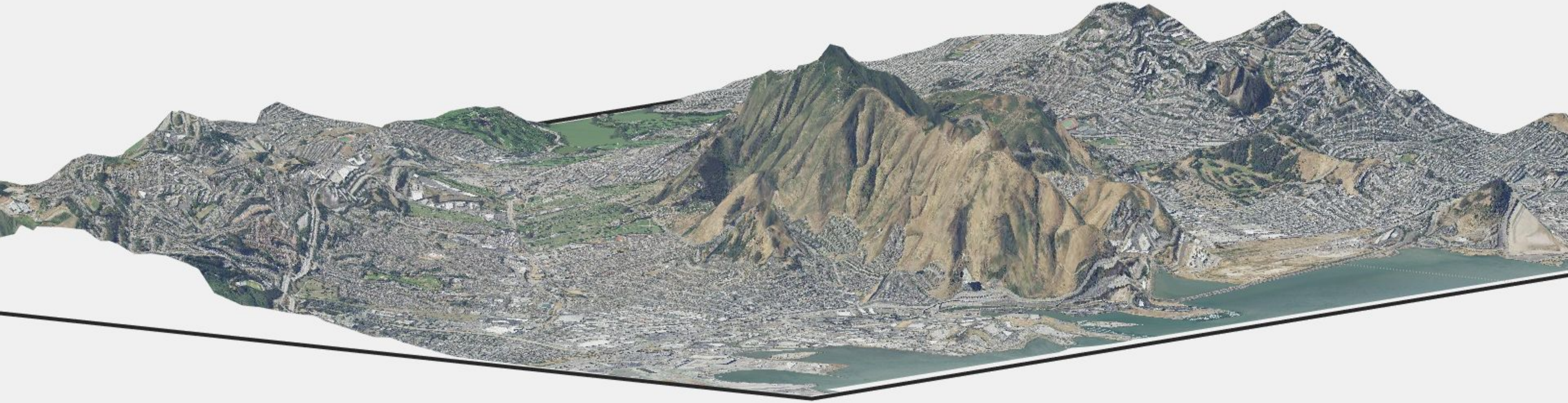
- Camera Manipulation
- Drone Path Generation
- Lighting

Terrain Data: Loading in DEMS

The process of loading in USGS DEM files can be broken into a couple steps

1. Loading in the DEM file and reading in the Latitudes, Longitudes, and Altitude Data.
2. From the data, find the min/max of the latitudes/longitudes so we can identify the bounds of the DEM data.
3. Making a request using our bounds to the Matlab WMS (Web Map Service) in order to get an orthographic satellite image of the area described by the DEM
4. Plotting the data using geoshow, and draping the orthographic image over the surface.

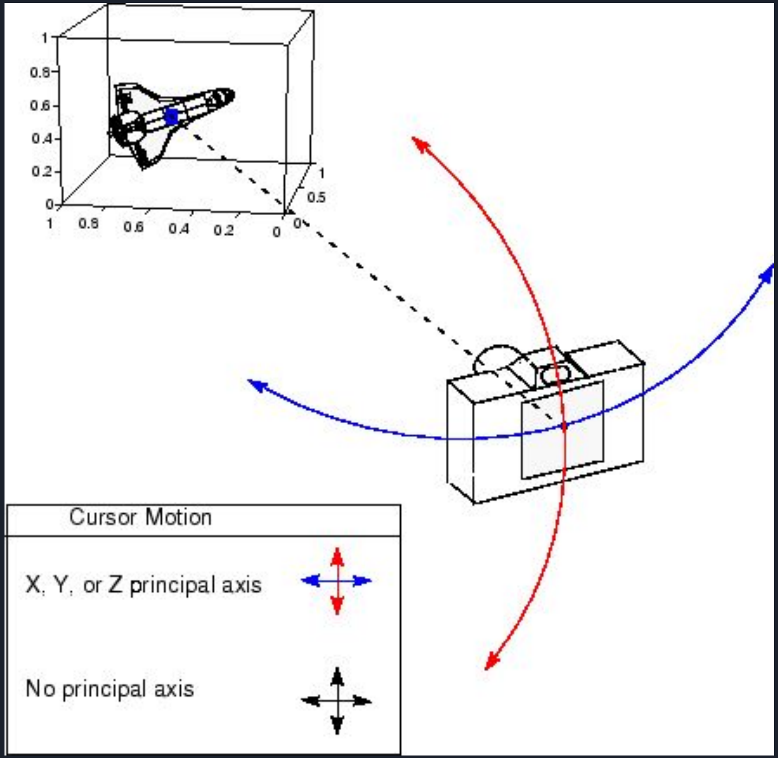
San Francisco Bay Area



Camera Position/View

- There are two components to manipulating the view of a surface in matlab
  - Camera Position
  - Camera Target
- We approached the topic of each of these separately, but with the same approach.
- We will use linear approximation to generate a linear function that will represent the position/target, moving from one location to another

Visualization of how the camera works in matlab



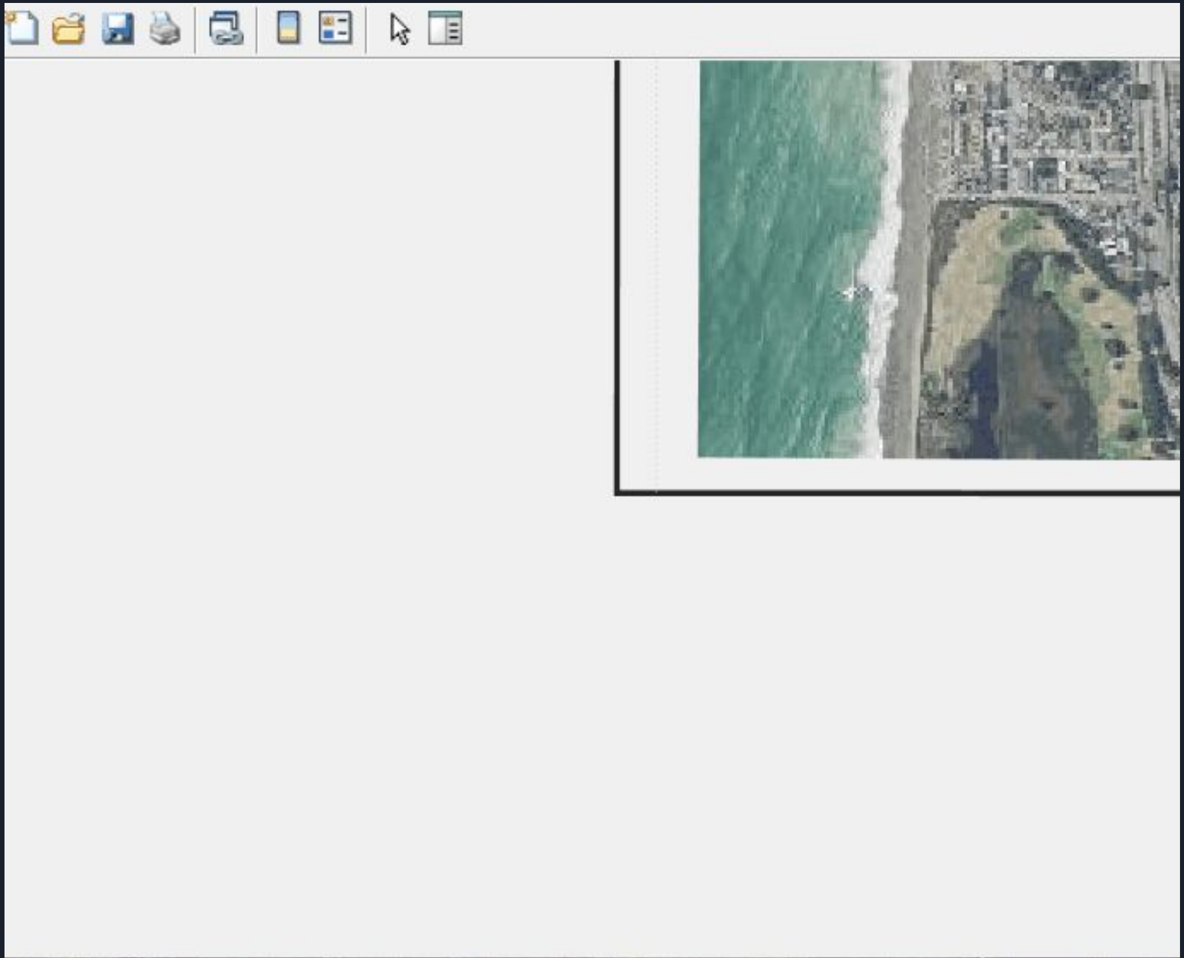
Basics - Moving in a Line

```
function Line(obj)
    obj.Display();
    [xm,ym,z]=camposm(min(obj.latlim),min(obj.lonlim),10000)
    [xM,yM,z]=camposm(max(obj.latlim),max(obj.lonlim),10000);
    l=linspace(xm,xM,100);L=linspace(ym,yM,100);zoom(1.25);
        for i=1:100
        camva(10)
        campos([l(i) L(i) 9000])
        camtarget([l(i) L(i) max(max(obj.Z))])
        drawnow
        pause(.05)
        end
end
```

Using:

- Camva
- Camposm
- Campos
- camtarget

Diagonal Straight Line

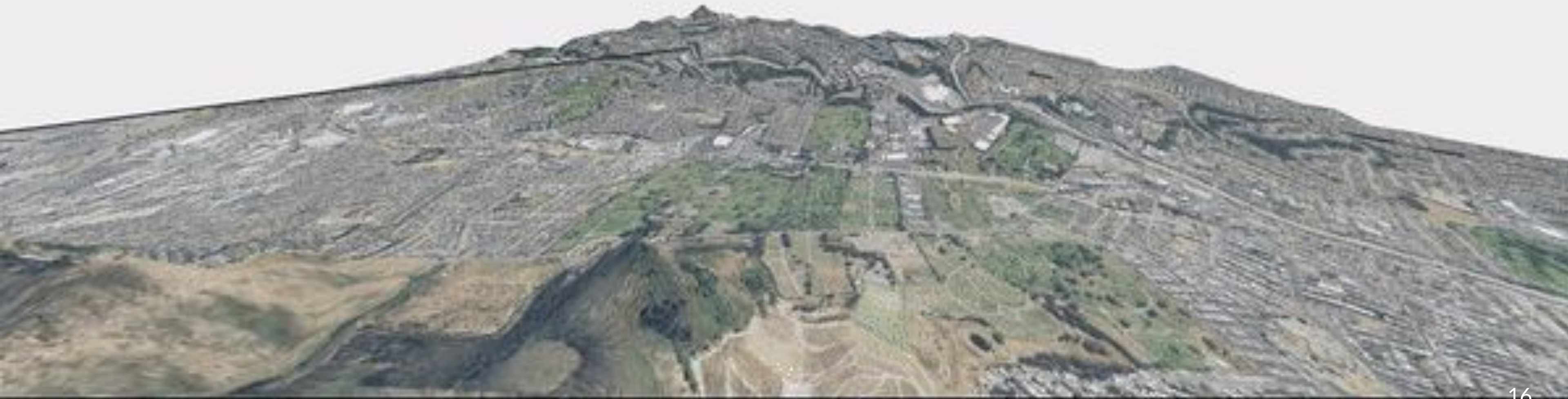


Looking in a Circle

```
function LookCircle(obj)
    obj.Display();
    [xm,ym,z]=camposm(min(obj.latlim),min(obj.lonlim),10000)
    [xM,yM,z]=camposm(max(obj.latlim),max(obj.lonlim),10000);
    l=linspace(xm,xM,50);L=linspace(ym,yM,50);zoom(1.5)
    camproj('perspective');k=0
        for i=1:99
            k=k+1
            camva(90)
            campos([mean(l) mean(L) max(max(obj.Z))+100])
            if k<50
            camtarget([l(1) L(k) max(max(obj.Z))])
            end
            if k>=50
            camtarget([l(k-49) L(50) max(max(obj.Z))])
            end
            pause(0.1)
            drawnow
        end
        k=0;
        for i=1:99
            k=k+1
            if k<50
            camtarget([l(50) L(50-k) max(max(obj.Z))])
            end
            if k>=50
            camtarget([l(100-k) L(1) max(max(obj.Z))])
            end
            pause(.1)
            drawnow
        end
end
```

- Set camera position of which we want to "place" the drone/camera
- Changing camera target based on a set timer (in this case 0.1 seconds)

View of San Francisco from the peak.



Linear Approximation.

- Obtain Longitude, Latitude, and Altitude data for two points X_1 and X_2
- Subtract X_2 and X_1 to get the slope vector (m)
- Have $v_0 = X_1$
- Return the slope (m) and initial position (b) vectors
- Use the form
  - $v(t) = m*t + v_0$, $0 <= t <= 1$
- And use loops to go through the period t to simulate the drone “flying” or camera “movement”

Camera Position

Using a set of data points (coordinates) to represent the camera position throughout the environment. We generate a set of linear equations for each “drone” position we want the camera to be at.

Using the Linear Approximation method described above we can seamlessly transition the camera from position, to position, over one path, or many paths

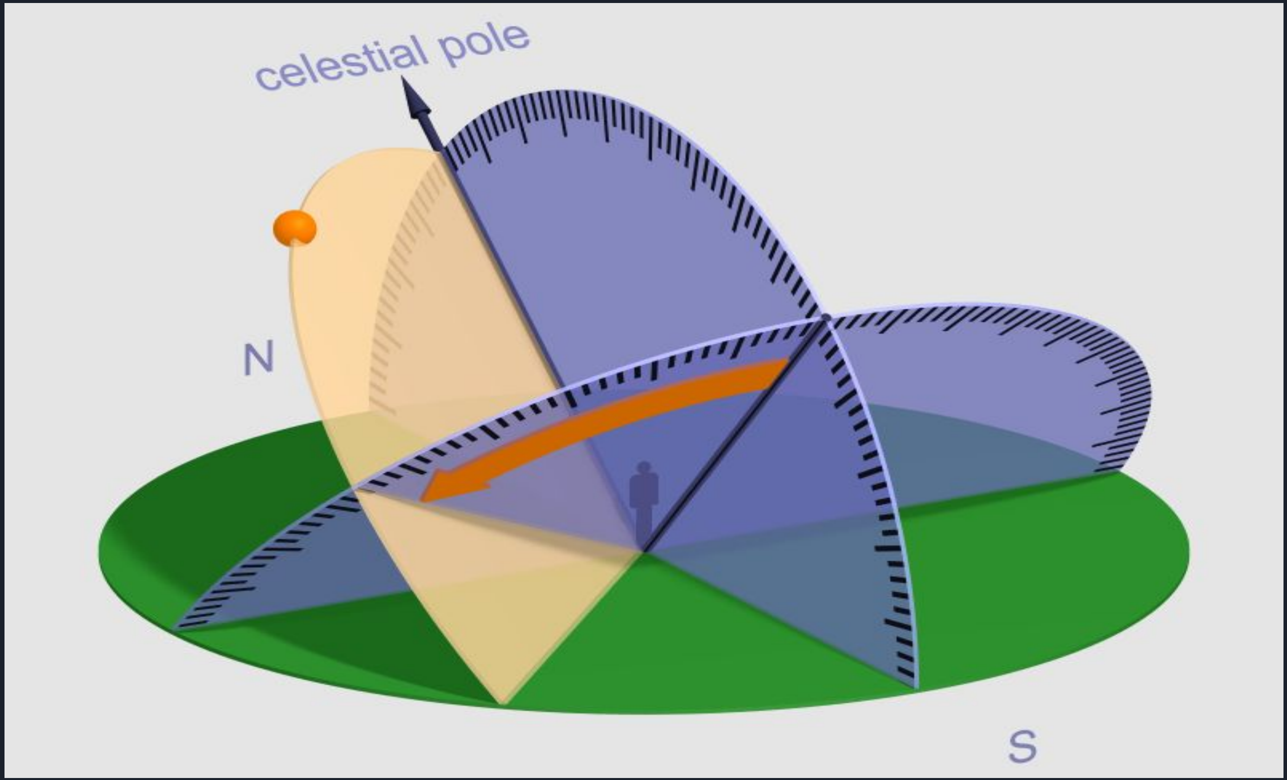
Camera Target

Using a set of data points (coordinates) to represent the camera target throughout the environment. We generate a set of linear equations for each target position we want to point the camera at.

Using the Linear Approximation method described above we can seamlessly transition the drone from position, to position, over one path, or many paths.

```
function [M,B] = get_linear_approx(X)
    M = zeroes(length(X)-1);
    B = M;
    for i = 1:length(X)-1
        [m,b] = get_line(X(i), X(i-1));
        M(i) = m;
        B(i) = b;
    end
end

function mag = magnitude(X)
    mag = sqrt(X.*X);
end

function [m,b] = get_line(x1,x2)
    m = x2-x1;
    b = x1;
end
```

```
function obj = fly_to(obj,dest,speed)
    X1 = obj.Current_Location;
    X2 = dest;
    [m,b] = get_line(X1,X2);
    T = linspace(0,1,100);
    for n = 1:length(T)
        V = m*T(n) + b;
        camposm(V(1),V(2),V(3));
        obj.view();
        pause(1/speed);
    end
    obj.Current_Location = X2;
end
```



The hour angle is indicated by an orange arrow on the celestial equator plane. The arrow ends at the hour circle of an orange dot indicating the apparent place of an astronomical object on the celestial sphere.

Sunrise/Sunset Algorithm

Inputs:

day, month, year: date of sunrise/sunset

latitude, longitude: location for sunrise/sunset

zenith: Sun's zenith for sunrise/sunset

offical = 90 degrees 50'

civil = 96 degrees

nautical = 102 degrees

astronomical = 108 degrees

```
1. first calculate the day of the year

        N1 = floor(275 * month / 9)
        N2 = floor((month + 9) / 12)
        N3 = (1 + floor((year - 4 * floor(year / 4) + 2) / 3))
        N = N1 - (N2 * N3) + day - 30

2. convert the longitude to hour value and calculate an approximate time

        lngHour = longitude / 15

        if rising time is desired:
          t = N + ((6 - lngHour) / 24)
        if setting time is desired:
          t = N + ((18 - lngHour) / 24)
```

```
3. calculate the Sun's mean anomaly

        M = (0.9856 * t) - 3.289

4. calculate the Sun's true longitude

        L = M + (1.916 * sin(M)) + (0.020 * sin(2 * M)) + 282.634
        NOTE: L potentially needs to be adjusted into the range [0,360) by adding/subtracting 360

5a. calculate the Sun's right ascension

        RA = atan(0.91764 * tan(L))
        NOTE: RA potentially needs to be adjusted into the range [0,360) by adding/subtracting 360

5b. right ascension value needs to be in the same quadrant as L

        Lquadrant  = (floor( L/90)) * 90
        RAquadrant = (floor(RA/90)) * 90
        RA = RA + (Lquadrant - RAquadrant)

5c. right ascension value needs to be converted into hours

        RA = RA / 15

6. calculate the Sun's declination

        sinDec = 0.39782 * sin(L)
        cosDec = cos(asin(sinDec))
```

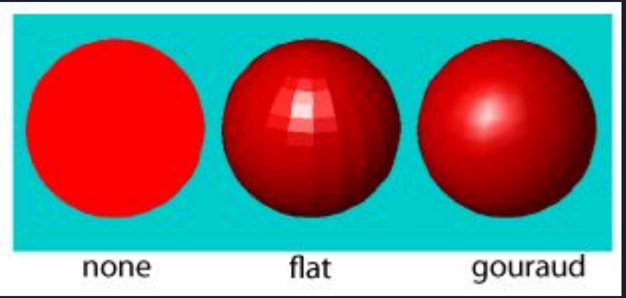
```
7a. calculate the Sun's local hour angle

	cosH = (cos(zenith) - (sinDec * sin(latitude))) / (cosDec * cos(latitude))

	if (cosH >  1)
	  the sun never rises on this location (on the specified date)
	if (cosH < -1)
	  the sun never sets on this location (on the specified date)

7b. finish calculating H and convert into hours

	if if rising time is desired:
	  H = 360 - acos(cosH)
	if setting time is desired:
	  H = acos(cosH)

	H = H / 15
```

Lighting

```
>> h=light

h =

  Light with properties:

        Color: [1 1 1]
        Style: 'infinite'
     Position: [1 0 1]
      Visible: 'on'

  Show all properties
```

Lighting

Make a light with lighting method gouraud.




```
h=light();
lighting gouraud
```

Lighting

```
h=light();
lighting gouraud
for i=1:99
    deg=i-9;
    az=0;
    lightangle(h,az,deg);
    sunsetlights(h);
    pause(.1)
end
```

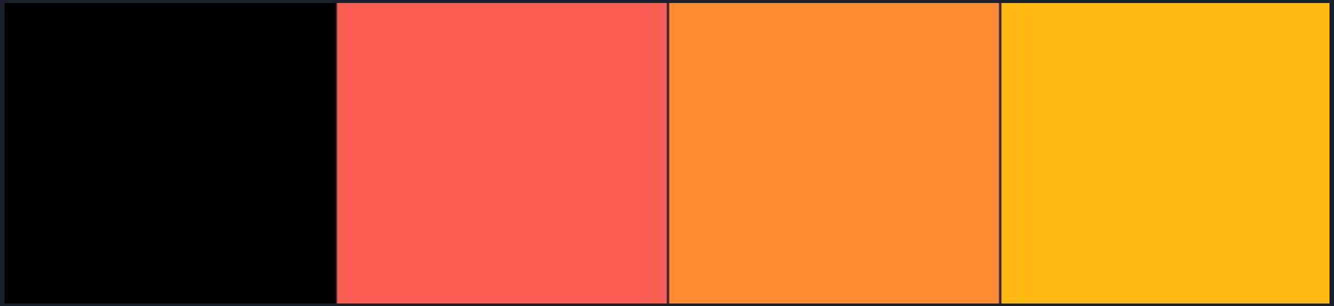
Lighting

```
function sunsetlights(h)
    redsunset=253;%variables
    yellowsunset=94;
    bluesunset=83;
    redsun=253;
    yellowsun=184;
    bluesun=19;
    sunsetanglechange=20;
    %function
    p=h.Position;
    r=sqrt(p(1)^2+p(2)^2);
    angle=atan(p(3)/r);
    red=253;
    yellow= min( yellowsunset+angle*(yellowsun-yellowsunset)/(sunsetanglechange*pi/180),yellowsun );
    blue= max( bluesunset+angle*(bluesun-bluesunset)/(sunsetanglechange*pi/180),bluesun);
    h.Color =[red/255 yellow/255 blue/255 ]
end
```

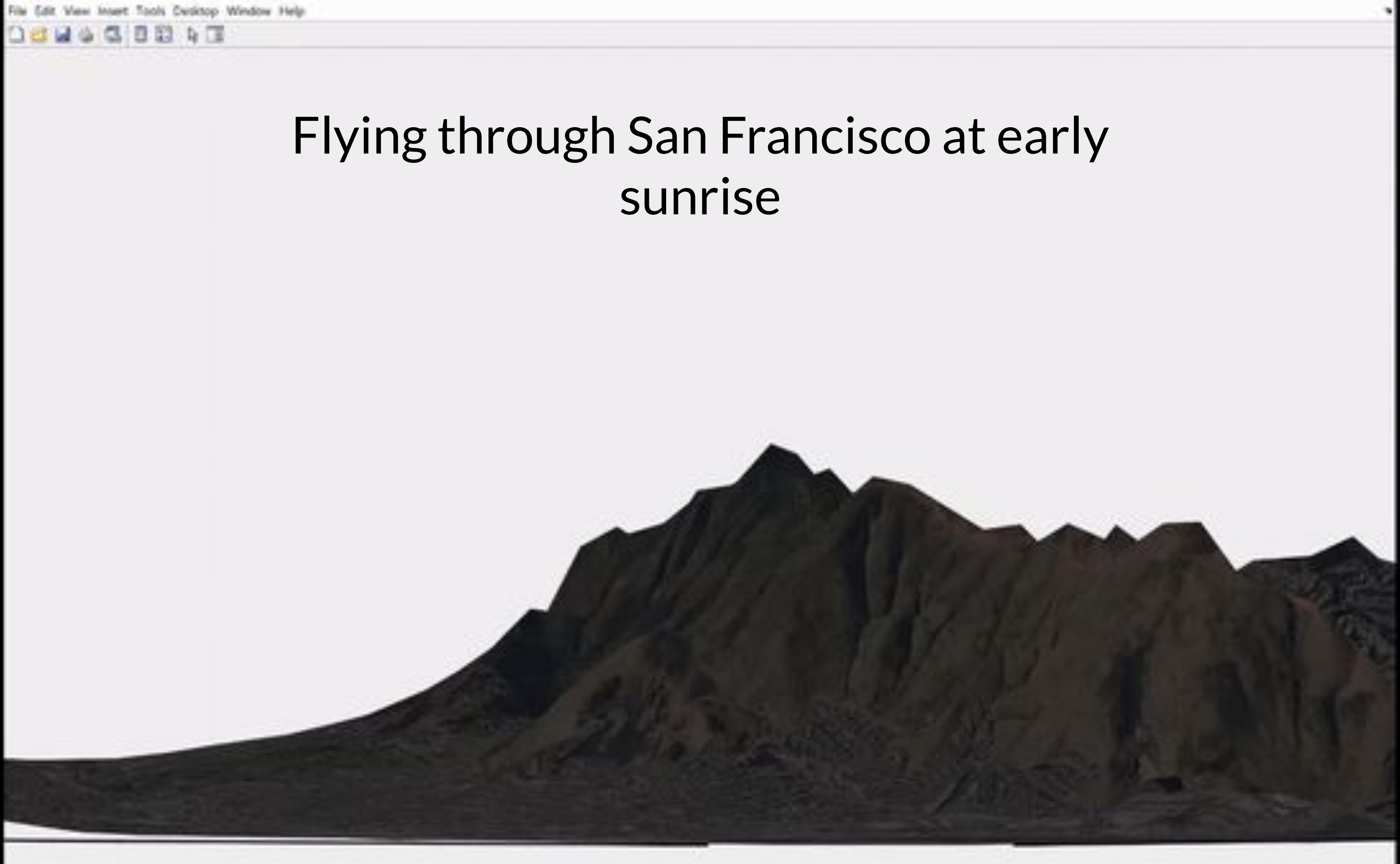
Lighting

Function Sunset Lights

<0 degrees | 0 degrees | 10 degrees | 20 degrees



Flying through San Francisco at early sunrise



Future Goals

- Collision Avoidance
- No-fly zones
- Energy Efficiency
- Flying Over Pluto

Conclusion

- Soaring over a terrain with a drone/camera.
- Camera view point
- Drone Flightpath
- Lighting effects on terrain over passage of time
- How these affect each other

Live Demo Time!

References

- [1] https://wingtra.com/drone-mapping-applications/surveying-gis/
- [2] D'Amato, E., Mattei, M. & Notaro, I. J Intell Robot Syst (2019) 93: 193. https://doi.org/10.1007/s10846-018-0861-1
- [3] https://en.wikipedia.org/wiki/Hour_angle#/media/File:HourAngle_Observer_en.png
- [4] https://edwilliams.org/sunrise_sunset_algorithm.htm
- [5] https://en.wikipedia.org/wiki/Sunrise_equation
- [6] https://earthsky.org/space/pluto-charon-new-horizon-flyover-video
- [7] https://www.nature.com/articles/d41586-019-02027-3
- [8] German Gramajo and Praveen Shankar, "An Efficient Energy Constraint Based UAV Path Planning for Search and Coverage," International Journal of Aerospace Engineering, vol. 2017, Article ID 8085623, 13 pages, 2017
- [9] http://stars.astro.illinois.edu/celsph.html
- [10 https://www.britannica.com/science/anomaly-astronomy#ref105659]