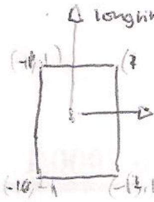
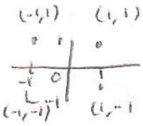
Car Physics

(operate in the direction of the car body)

- Need to consider longitudinal and lateral forces separately



- longitudinal forces (wheel force, braking force, rolling force and air resistance) = (0,1) y direction
- lateral forces (allow car to turn, angular moment and tork). = (1,0) x direction



Newtons second law of motion $F = MA$ (mass is constant)

for longitudinal

$A = F/M$

$V = V + A*T$

$P = P + V*T$

V = linear velocity

T = time step

P = car position

for lateral

AngA (~~Angular Acceleration~~) $= Te/I$

AngV (~~Angular velocity~~) $= AngV + AngA*T$

$Angle = Angle + AngV * T$

AngA = angular acceleration

T = torque

I = inertia

AngV = angular velocity

very similar

- Buildings JSON file ~~Feature Collection~~

```
{ {
(1) "type" = "Feature Collection",
(1.1) "features" : [ {
  (1.1.1) "geometry" : {
    (1.1.1.1) "type": "Polygon",
    (1.1.1.2) "coordinates" : [
        [   [x1,y1],
            [x2,y2],
            [x3,y3],
            [x4,y4]
        ]
    ],
  (1.1.2) "type" = "Feature"
  (1.1.3) "id" : "ac5c"
  (1.1.4) "properties" : {}
  }, etc.
 }]
}
```

```
(1.1.1) public class Geometry {
  (1.1.1.1) public String type { }
  (1.1.1.2) public ArrayList<float> coordinates { }
}

(1.1.4) public class Properties {

}

(1.1) public class Feature {
  (1.1.1) public Geometry geometry { }
  (1.1.2) public String type { }
  (1.1.3) public String id { }
  (1.1.4) public Properties properties { }
}

public class Root Object {
  (1) public String type { }
  (1.1) public ArrayList <Feature> features { }
}
```

– probably should include getters and setters for these

[[[x,y] [x1,y2]]]

length of equator = 40075.016686 KM
40075016.686 m

Horizontal tile at zoom 0 = 156543.034 m

Distance rep by 1 pixel = $\frac{C * \cos(y)}{2^{(z+8)}}$

C = equitorial circumference

z = zoom level

y = latitude

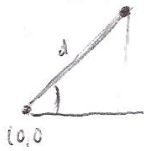
Long lat to world conversion

- have formula to find distance between 2 points
- Pick one point which is fixed (use as center) and find distance between all other points

- x = long y = lat

longitude ↑

latitude →



1 2 3

4 1 1

12 34 56

0 1 2

0 1 23 45

Sphere at centre 0,0

$x = R * \cos(lat) * \cos(lon)$

$y = R * \cos(lat) * \sin(lon)$

$z = R * \sin(lat)$

$lat = \arcsin(z/r)$

$lon = \arctan(y, x)$

~~then project onto parallel~~

get x, y and z, then use parallel projection to plot 3D points onto 2D.

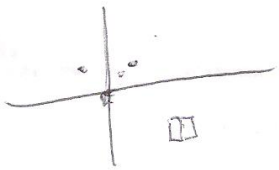
~~$Q_x = \cos(lat) * \cos(lon)$~~

look at mercater projection

-4129.5

122 122 - -122

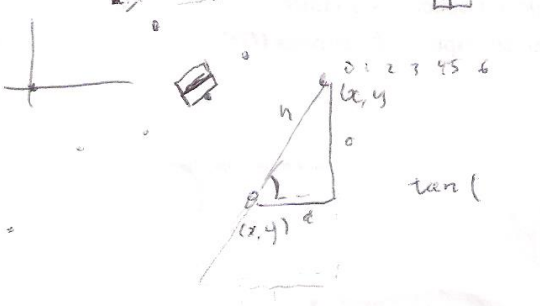
122

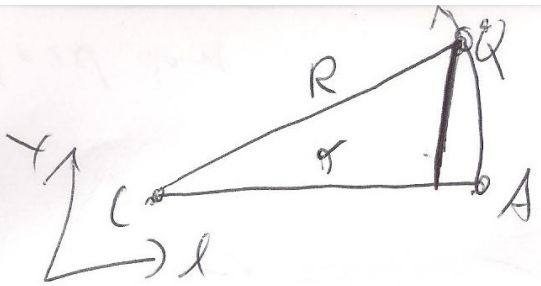


need to parse geoJSON properly

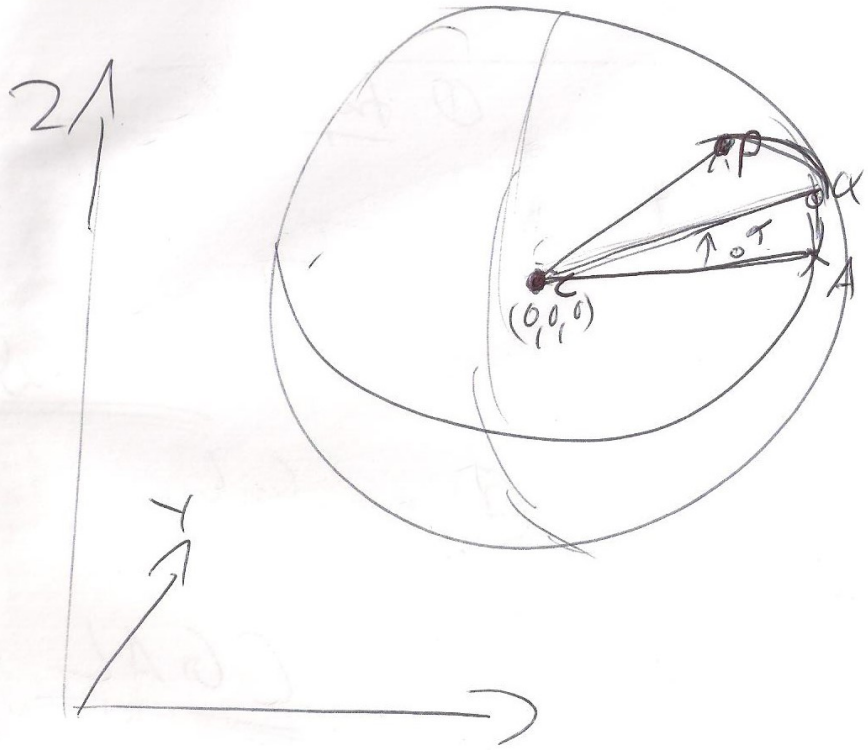
[[[,]]]

[[ ], [ ], [ ]]





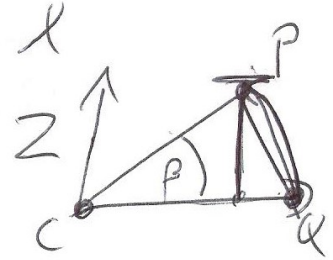
$R = 1$



$$X = C_x + R\cos\alpha \cos\beta$$

$$Y = C_y + R\sin\alpha \cos\beta$$

$$Z = C_z + \sin\beta$$



→ polar coordinates

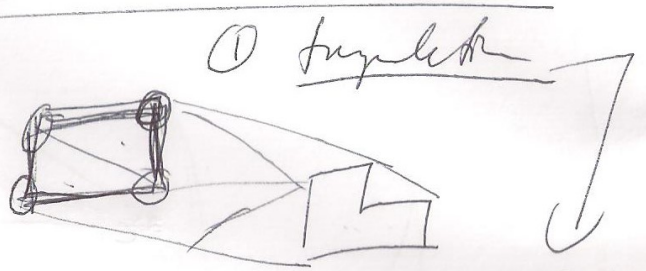
→ generally maps 3D → 2D projections

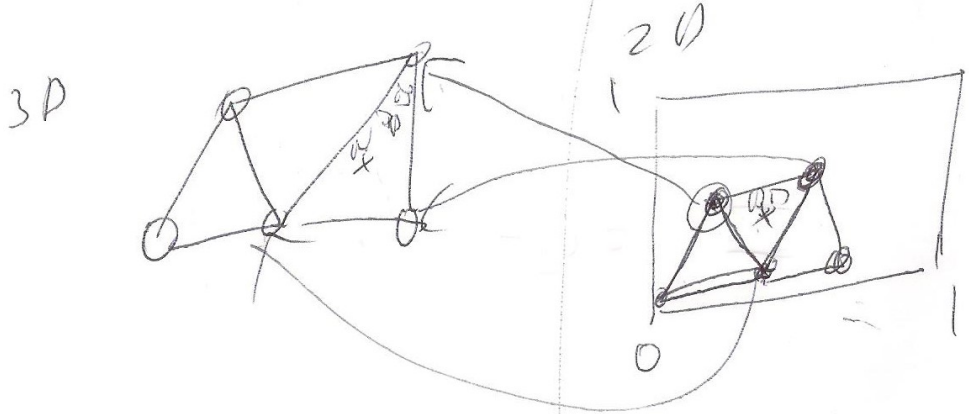
conformal : angle - preserv

equiareal ~~angle~~ area - preserv

map projections

mesh parameterization





② parameterize
mesh

OpenMesh

CGAL

OBJ → req. format

- Target Resolution - 480 x 800 (typical Android handset) - 800 x 480 (in landscape mode)
- Aspect ratio - 1.67
- 32 pixels = 1 metre, maybe 20pixels = 1 metre, or 24pixels = 1 metre (typical car is 4 metres, car should be 96pixels long on screen ∴ 96/4 = 24)
- 20 pixels = 1 metre - car is 90pixels ~~in length~~ / ~~4 metres~~ 4.5 metres in length, ∴ 20 pixels per metre

4.5m length of car

Abstract Screen

World controller

Wheels

4

Car

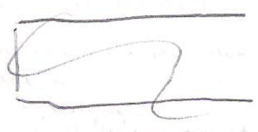
~~get velocity~~

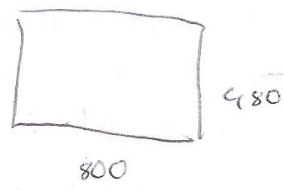
width, float
length, float
angle float
power, float
maxSpeed float
Vector2 position

96

45

Tuesday 11th 10.00-30n





Cutycapt

brightfuture ~~[illegible]~~

01242236415

Stevie

archiva

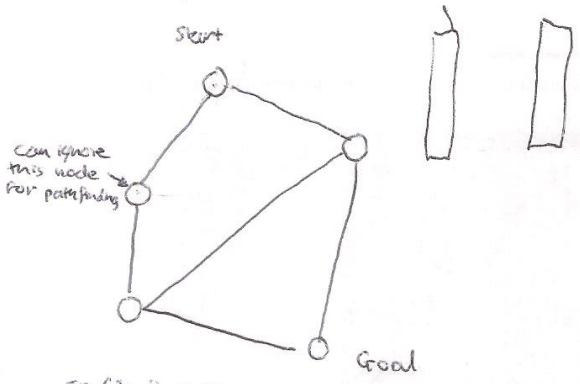
Meeting with Frank

Major Problem

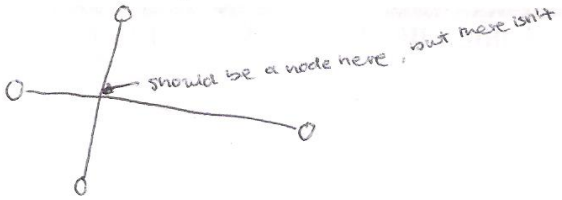
RoadNode

Vector2 ~~startpoint~~
~~Vector2 endpoint~~
RoadNode[] adjacentNodes

RoadLine
~~Vector2 startPoint~~
~~Vector2 endPoint~~
RoadNode startPoint
RoadNode endPoint



In file it is like this



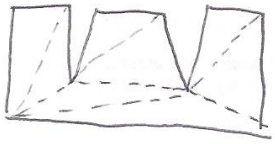
- Sweep line algorithm, Bentley ~~Ottmann~~, ~~Boolean~~

Either find intersection and create node or create own file server that returns ~~raw data~~ all the required data, see File Stache

Minor Issues

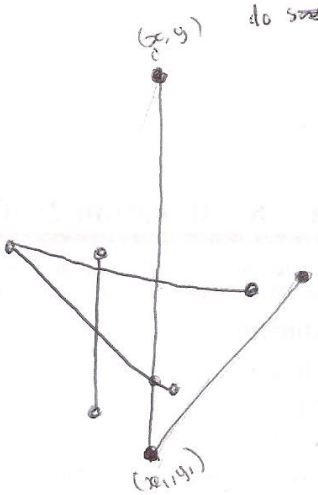
- BoxCD doesn't support concave polygons e.g.

Need to partition object into polygons (triangulation).

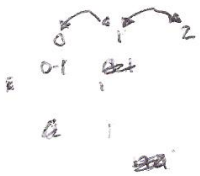


= convex Hull ~~problem~~ algorithm

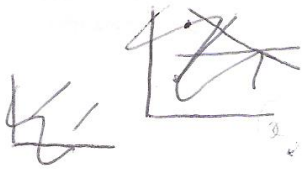
- Bridges, have ignored them for the time being

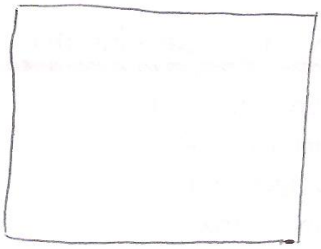


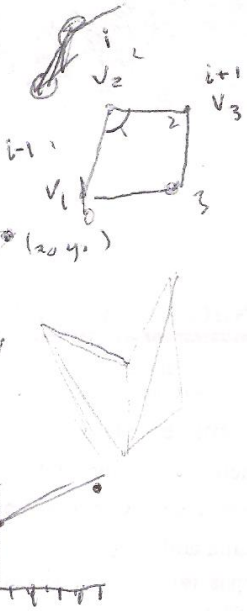
do bentley ottman before creating nodes



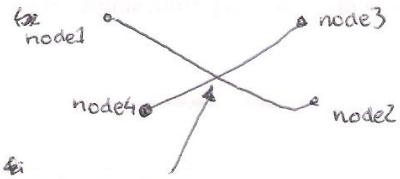
nodes lines can be



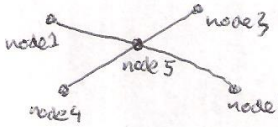




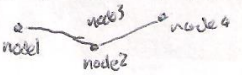
Line Intersection Algorithm



- Find point of intersection
- Search all nodes to it there is anything there
- If there is nothing there create a new node with pointers to all other nodes

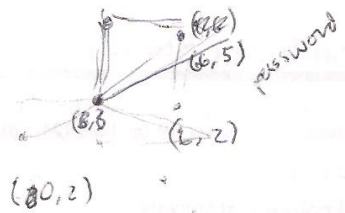


- Ignore if nodes are disjoined at points e-g.



- Assume that no 2 nodes can have the same position unless it is a bridge

host-name nabeel.sizyserver.com



(0,2)

(6,5)

(1,2) $y = mx + c$

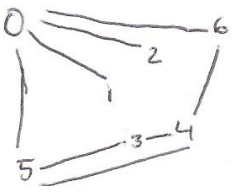
```
public class Edge {
    private Node a,b;
}
```

```
public class ~~Toe~~ Node {
    private Vector2 point
    ~~private int index~~
    private int isBridge
}
```

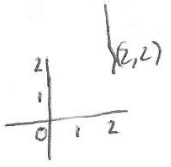
use hash table

Edges represent streets, nodes represent crossings

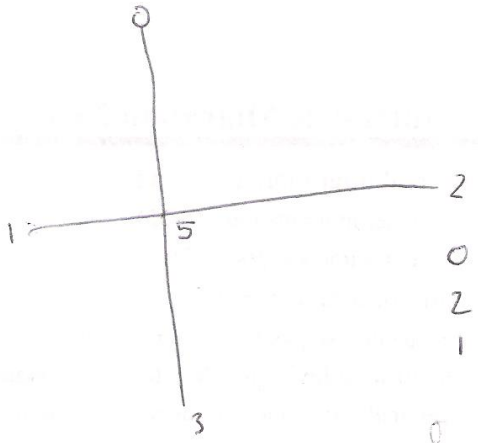
- Ignoring one way roads, ~~all roads are link~~ so we have a ~~directed~~ undirected graph, ~~nodes are connected to adjacent~~
Edge (a,b) is equal to (b,a). Could store as linked list or adjacency matrix. Use adjacency-set graph representation



0: {1, 2, 5, 6}
1: {0}
2: {0}
3: {4, 5} etc



(0,0) (2,2) (2,0) (0,2)

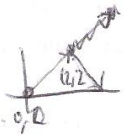


0 2
2 1
1 4

Intersection

$(x_1 - x_2)(y_3 - y_4) - (y_1 - y_2)(x_3 - x_4) = 0$

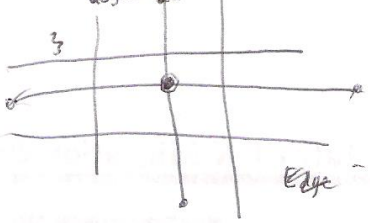
if lines are parallel



(0,2) (0,2) (0,0) (2,2)

Node {
point
}

~~Edges~~
AdjacentNodes {
point
~~adjacent~~ point array
}

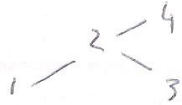


algorithm recognises ~~these~~ when lines are joined need to toe

0 - 2
0 - 3

Road Node

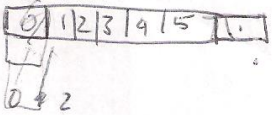
1 2
2 4
2 3



Edge
~~Vertex~~
Node
Adjacent N

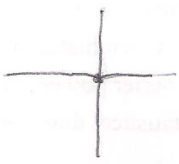
	0	1	2	3	4
2	3	0	0	1	
3	4				

~~Matrix~~



1 0 1
1 0

[1, 2, 3 4]

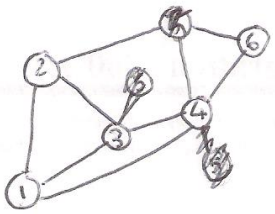


-1055.514 -144.32647

1, 3
1 4
2 3
2 4

Graph Traversal Algorith BFS

Goal to get from 1 to 6



adjacencylist

1 = {2, 3, 4}
2 = {1, 3, 5}
3 = {1, 2, 3}
4 = {1, 3, 5, 6}
6 = {5, 4}

path to goal with BFS

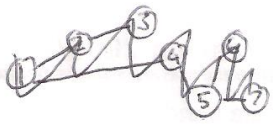
frontier {1} ~~expand~~ get all nodes from 1 closed {1} ← added 1 to closed list

frontier {2, 3, 4} ~~closed {1}~~ closed {1, 2, 3, 4}

frontier {3, 4, 5} closed {1, 2, 3, 4, 5}

frontier {4, 5} closed {1, 2, 3, 4, 5}

frontier {6, 5}
↑ goal found



Get Road tiles

- Select lat, long
- Convert to x, y, 200m
- Get tiles either side of it and up and down

~~(x+1)~~ x+1, y, 200m
x, y+1, 200m
x+1, y+1, 200m
x-1, y, 200m
x, y-1

0	1	2
x-1, y-1, z	x, y-1, z	x+1, y-1, z
x-1, y, z	x, y, z	x+1, y, z
x-1, y+1, z	x, y+1, z	x+1, y+1, z

WX 3.07

WX 3.14

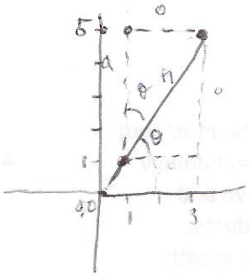
0 1 2 3

Category has subcategories
id
name
etc

store in array like this

categories = [ ]

~~category~~ category, name, id ... etc., parentid

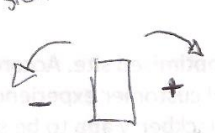


x = 3 - 1,
y = 5 - 1

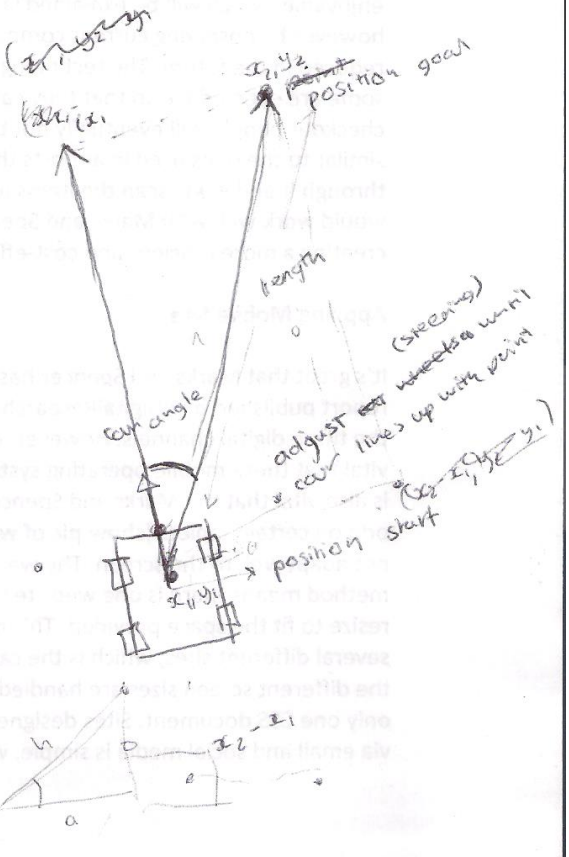
x = 2, y = 4

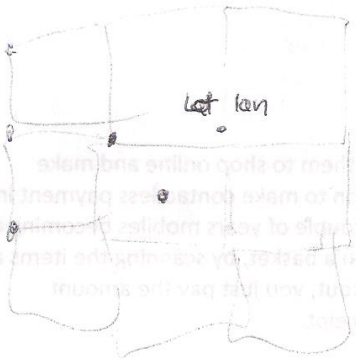
$\tan = \frac{o}{a}$

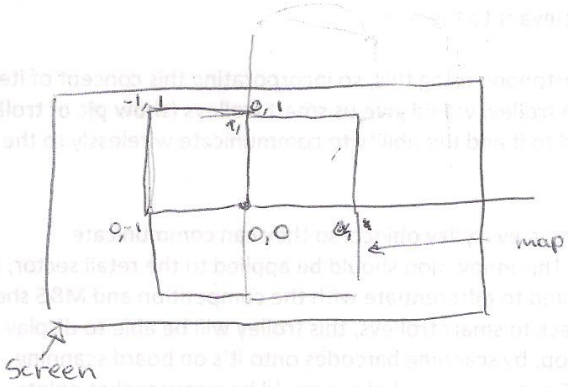
steer left steer right



$\frac{\text{x position of object}}{\text{magnitude of vector}}$ – gives value between −1 and 1







0,0 → -1,-1 1,0 → 0,1

0,1 → 0,-1 1,1 → 0,0

0,0	0,0
0,0	1,1