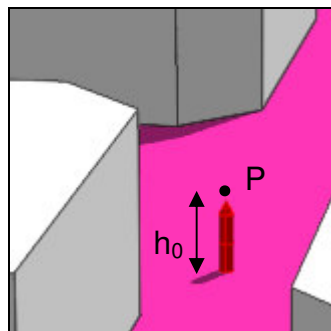
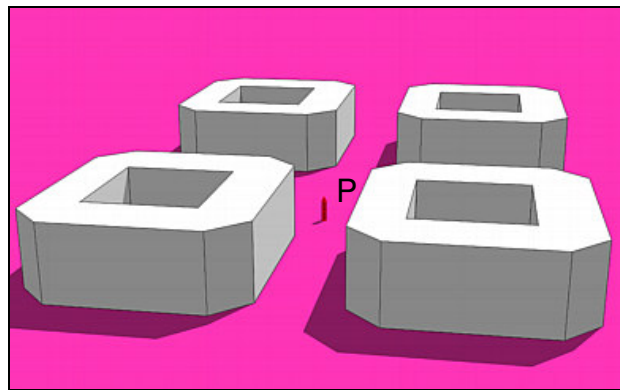


Capítol 5: Traspàs d'skylines, metodologia.

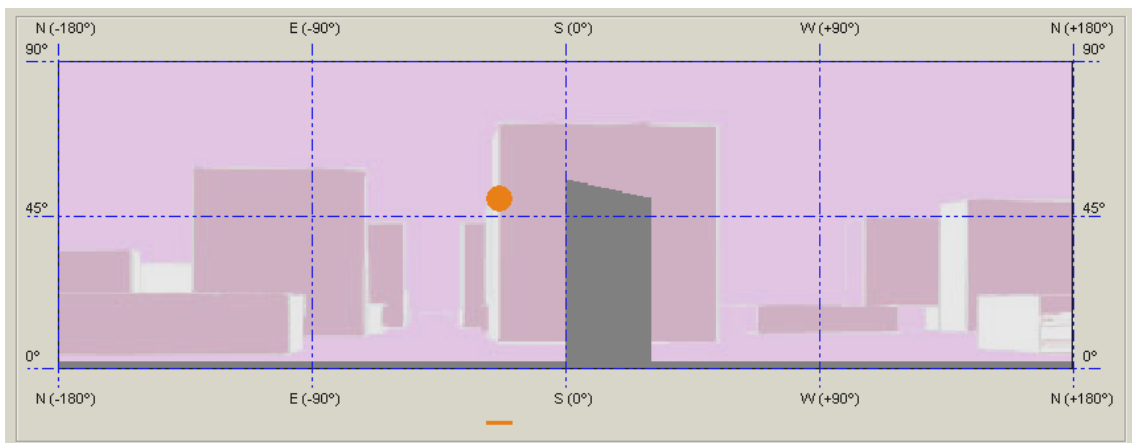
Resum

En aquest capítol es descriu l'estratègia per a definir un skyline a partir de la descripció en 2D de les arestes dels edificis o obstacles adjacents.

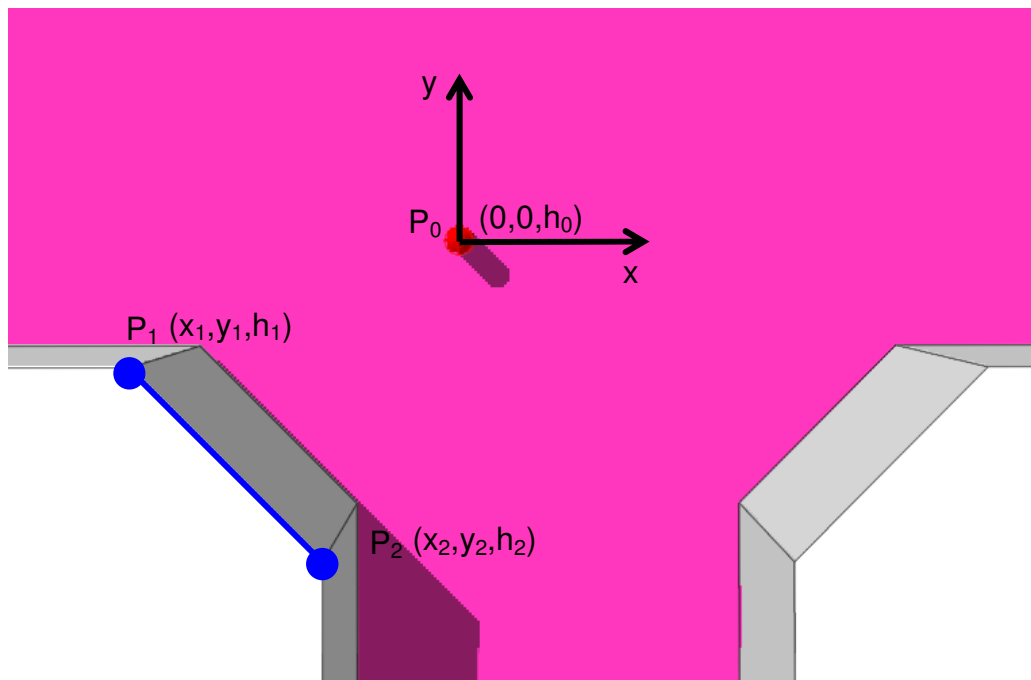
Es té una volumetria com la següent, i es vol traspasar la informació dels diferents obstacles (definit per la seva desviació respecte al Sud i altura solar) de un punt qualsevol P a una altura h_0 .



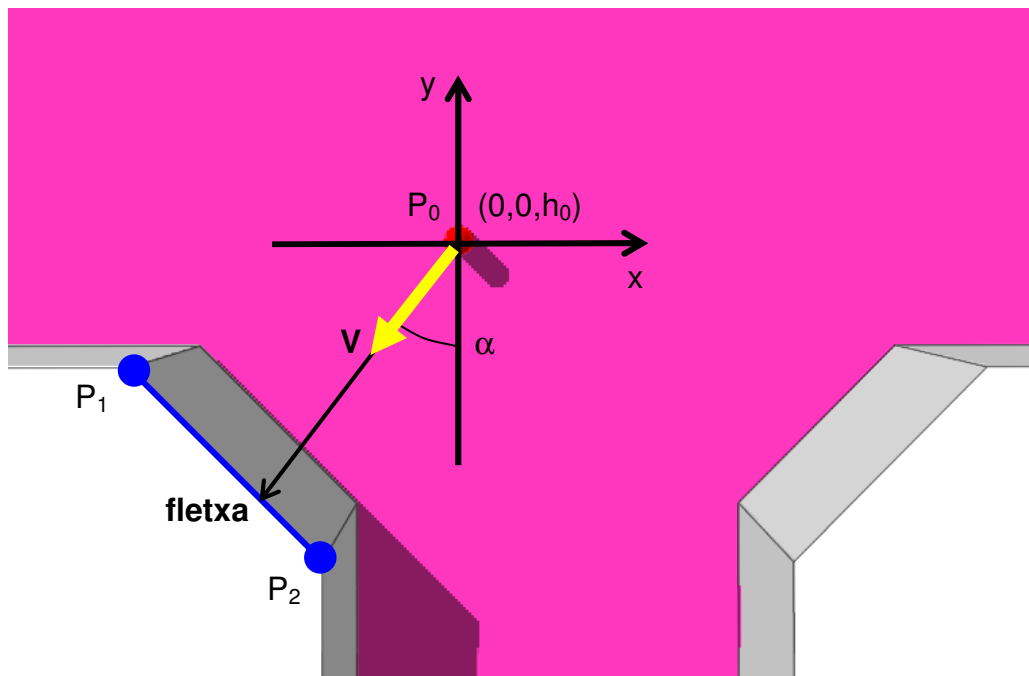
L'objectiu es passar els obstacles visuals que representen les arestes més elevades dels edificis adjacents a un mapa 2-D on s'hi representa l'skyline a partir de la desviació i altura solar.



Estat de la situació per una franja d'obstacle definida pels seus dos extrems $P_1(x_1, y_1, h_1)$ i $P_2(x_2, y_2, h_2)$ i visualitzada desde $P_0(0, 0, h_0)$



Es tracta de veure ara, per a una direcció determinada (fletxa apuntadora desde P_0), si existeix interferència visual amb el segment P_1, P_2 , i si existeix a quina altura es dona.



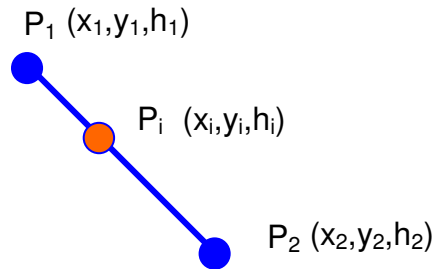
El vector unitari en la direcció α (desviació respecte al Sud) és V:

$$\vec{V} = [-\sin(\alpha) - \cos(\alpha)]$$

i la projecció d'aquest vector és en principi:

$$\text{fletxa} = \eta[-\sin(\alpha) - \cos(\alpha)]$$

El segment (P1,P2) es defineix de forma paramètrica (molt millor pels propòsits de representació posterior) com a:



$$P_i = [x_i, y_i, h_i] = [x_1, y_1, h_1] + \lambda[x_2 - x_1, y_2 - y_1, h_2 - h_1] \text{ amb } 0 < \lambda < 1$$

La intersecció en (2-D) entre la fletxa i el segment (si es dona) ve donada per:

$$\left. \begin{array}{l} \text{Fletxa:} \quad [x_i, y_i] = \eta[-\sin(\alpha) - \cos(\alpha)] \\ \text{Segment} \quad [x_i, y_i] = [x_1, y_1] + \lambda[x_2 - x_1, y_2 - y_1] \end{array} \right\}$$

Les condicions que s'han de donar per què es creuin el segment i la fletxa són:

- $\eta > 0$ (en el sentit en què apunta la fletxa)
- $0 < \lambda < 1$ (s'intersecciona dins del segment)

Equació d'intersecció (vectorial)

$$-\eta[\sin(\alpha), \cos(\alpha)] = [x_1, y_1] + \lambda[x_2 - x_1, y_2 - y_1]$$

Es descomposa en:

$$\left. \begin{array}{l} -\eta \cdot \sin(\alpha) = x_1 + \lambda(x_2 - x_1) \\ -\eta \cdot \cos(\alpha) = y_1 + \lambda(y_2 - y_1) \end{array} \right\}$$

I les solucions

Per la lambda (λ):

CAS 1)

si $\tan(\alpha) \cdot (y_2 - y_1) - (x_2 - x_1) \neq 0$

i $\alpha \neq 90^\circ$ ó $\alpha \neq -90^\circ$

$$\lambda = \frac{x_1 - y_1 \cdot \tan(\alpha)}{\tan(\alpha) \cdot (y_2 - y_1) - (x_2 - x_1)}$$

CAS 2)

si $\tan(\alpha) \cdot (y_2 - y_1) - (x_2 - x_1) = 0$

$$\lambda = \text{inf}$$

CAS 3)

si $\alpha = 90^\circ$ ó $\alpha = -90^\circ$

si $y_2 \neq y_1$

$$\lambda = -\frac{y_1}{(y_2 - y_1)}$$

CAS 4)

si $\alpha = 90^\circ$ ó $\alpha = -90^\circ$

si $y_2 = y_1$

$$\lambda = \text{inf}$$

Per la rho (η):

agafarem $\eta = -\frac{x_1 + \lambda(x_2 - x_1)}{\sin(\alpha)}$ si $[\alpha \neq 0]$ ó $[\alpha \neq 180^\circ]$

sinó $\eta = -\frac{y_1 + \lambda(y_2 - y_1)}{\cos(\alpha)}$ si $[\alpha = 0]$ ó $[\alpha = 180^\circ]$

Programació

A la pantalla

Altura del punt de referencia $h_0 = 20$

Desviació respecte al Sud: **desvsud**

M vectors des de $s = 0$ fins a M

S	x1(s)	y1(s)	h1(s)	x2(s)	y2(s)	h(s)
0	100	200	30	200	200	30
1	50	20	40	20	20	40
2
M	60	100	20	-60	-60	20

Definim el vector on posarem les altures que anem trobant per a cada segment

Altura(360): altura(0)altura(359)

el valor altura(i) representa l'altura a l'azimut (i-180)

el valor altura(0) -----) dona l'altura de l'azimut -180

el valor altura(1) -----) dona l'altura de l'azimut -179

.
. .
. .

el valor altura(359)-----) dona l'altura de l'azimut 179

ALGORITME

‘definició de variables

```
dim error as integer
dim beta, altura(360) as double
dim x1(20), y1(20),h1(20),x2(20),y2(20),h2(20) as double
dim ho, desvsud as double
dim alfa, lambda, rho, tol as double
Const pi = 3.141592653
Const grarad = 0.01745329252
Const radgra = 57.2957795129
tol = 0.001
```

‘primer inicialitzo el vector altura

```
for i =0 to 359
    altura(i) =0
next i
```

‘carrego els valors de h0, desvsud

```
ho = Cdbl(text1.text)
desvsud = Cdbl(text2.text)
```

‘carrego els valors del formulari cap a les variables que he definit, així treballa millor

```
for i= 0 to 19
    x1(i) = Cdbl (xx1(i).text)
    y1(i) = Cdbl (yy1(i).text)
    h1(i) = Cdbl (hh1(i).text)
    x2(i) = Cdbl (xx2(i).text)
    y2(i) = Cdbl (yy2(i).text)
    h2(i) = Cdbl (hh2(i).text)
next i
```

```
for s =0 to 19
  'inicialitzo el control d'error a zero, s'activa a 1 si el denominador del calcul de lambda = 0
  error = 0
  if indicador(s).value = true then
    'calcular lambda i rho per cada angle
    for i=0 to 359
      alfa = i-180
      'CALCUL DE LA LAMBDA
      'posar filtres per no fer operacions no vàlides
      '(cas 1) que tan(alfa) ≠ infinit i denominador ≠ 0
      if alfa ≠ -90 and alfa ≠ 90 then
        if abs[(y2(s)-y1(s)).tan(grarad.alfa)-(x2(s)-x1(s))]> tol then
          lambda =[x1(s)-y1(s). tan(grarad.alfa)]/[(y2(s)-y1(s)).tan(grarad.alfa)-(x2(s)-x1(s))]
        end if
      end if
      '(cas 2) que tan(alfa) = infinit i y1≠y2
      if alfa = 90 or alfa = -90 then
        if abs[(y2(s)-y1(s)) > tol then
          lambda =[-y1(s)]/[y2(s)-y1(s)]
        end if
      end if
      '(cas 3) que tan(alfa) = infinit i y2 = y1
      if alfa = 90 or alfa = -90 then
        if abs[(y2(s)-y1(s))≤ tol] then
          lambda = 1000
          error = 1
        end if
      end if
    end if
  end if

```

‘cas 4) denominador proper a zero amb $\tan(\text{alfa}) \neq \text{infini}$

```
if alfa  $\neq$  -90 and alfa  $\neq$  90 then  
    if  $\text{abs}[(y2(s)-y1(s)).\tan(\text{grad.alfa})-(x2(s)-x1(s))] \leq \text{tol}$  then  
        ‘li donem un valor elevat: vol dir que talla més enllà del segment  
        lambda = 1000  
        error =1  
    end if  
end if
```

‘CALCUL DE LA RHO

```
if alfa  $\neq$  0 and alfa  $\neq$  -180 then  
    rho =  $-[x1(s)+ \text{lambda} \cdot (x2(s)-x1(s))]/\sin(\text{grad.alfa})$   
else  
    rho =  $-[y1(s)+ \text{lambda} \cdot (y2(s)-y1(s))]/\cos(\text{grad.alfa})$ 
```

```
end if  
‘si rho >0 i  $0 < \text{lambda} < 1$  calcular l’altura solar  
if rho >0 then  
    if lambda >0 and lambda <1 then  
        beta =  $\text{radgra. atan} [(h1(s)+(h2(s)-h1(s)).\text{lambda}-h0)/ \text{rho}]$   
        ‘si l’altura calculada es superior a l’existent, substitueixo  
        if altura(i) < beta then  
            altura(i) = beta
```

```
    end if
```

```
    end if
```

```
end if
```

```
next i
```

```
end if
```

```
next s
```

‘trobar les altures al mòdul skylines

for i =0 **to** 359

 azimut = i-180

 azimutreal = azimut - desvsud

if azimutreal < -180 **then**

 azimutreal = azimutreal + 180

end if

if azimutreal > 180 **then**

 azimutreal = azimutreal - 180

end if

‘executo azimut i altura al mòdul dels skylines

if azimutreal <170 **then**

 formskyline.aziminf = azimutreal

 formskyline.azimsup = azimutreal +0,1

else

 formskyline.aziminf = azimutreal - 0,1

 formskyline.azimsup = azimutreal

end if

 formskyline.alt = altura(i)

 formskyline.executaelevar = true

next i