

Pràctica 1 DAT

En aquesta primera pràctica de DAT, veurem una introducció a **Haskell** i la **programació funcional**.

Per poder-la iniciar ens connectem per ssh al laboratori de DAT:

```
ssh -p 6767 ldatusrXX@soft00.upc.edu
```

Un cop dins necessitem crear i editar l'arxiu `.bash_profile` amb l'editor de textos q vulgueu, jo he fet servir nano, i afegim la següent línia:
`PATH="$PATH:~WEBprofe/usr/bin"`

Llavors executem `source .bash_profile` i ja podem iniciar les pràctiques.

Pràctica

Creem el directori de les pràctiques `cd practiques` i el de la `prac1` dins d'aquest.

Per executar les pràctiques utilitzarem l'scrip `WEBprofe/usr/bin/run-main` que compila i executa un programa Haskell tq:

```
$ run-main fitxer_codi.hs > sortida.svg
```

La sortida del programa es guarda en el fitxer `sortida.svg` i la podreu visualitzar amb qualsevol navegador que suporti SVG.

Primera part

Creem un fitxer haskell anomenat `myDrawing.hs` amb el següent codi:

```
import Drawing
```

```
myDrawing :: Drawing
myDrawing = blank
```

```
main :: IO ( )
main = svgOf myDrawing
```

l el compilem i executem així: `run-main myDrawing.hs > ~/public_html/prac1/ex0.svg`

Per visualitzar-lo introduïm la següent URL al nostre navegador: <http://soft0.upc.edu/ldatusrXX/practiques/prac1/ex0.svg> amb el nostre usuari de dat en comptes de les XX.

Modifiquem el programa canviant `myDrawing = solidCircle 1` per veure un cercle de color negre i radi 1.

Com que Haskell treballa amb programació funcional, si ara fessim una altre declaració de `myDrawing`: `myDrawing = solidCircle 2`, el compilador ens retornaria un error ja que en Haskell `=` no significa assignació, sinó definició, i per tant només podem definir una significat a cada variable.

Finalment editem el color del cercle, buscant a la documentació deñ mòdul `Drawing` trobem com es fa:

```
myDrawing = colored green (solidCircle 1) i ens dibuixa un cercle de color verd.
```

Exercici 2

Realitzem un programa que dibuixi un semàfor:

```
import Drawing

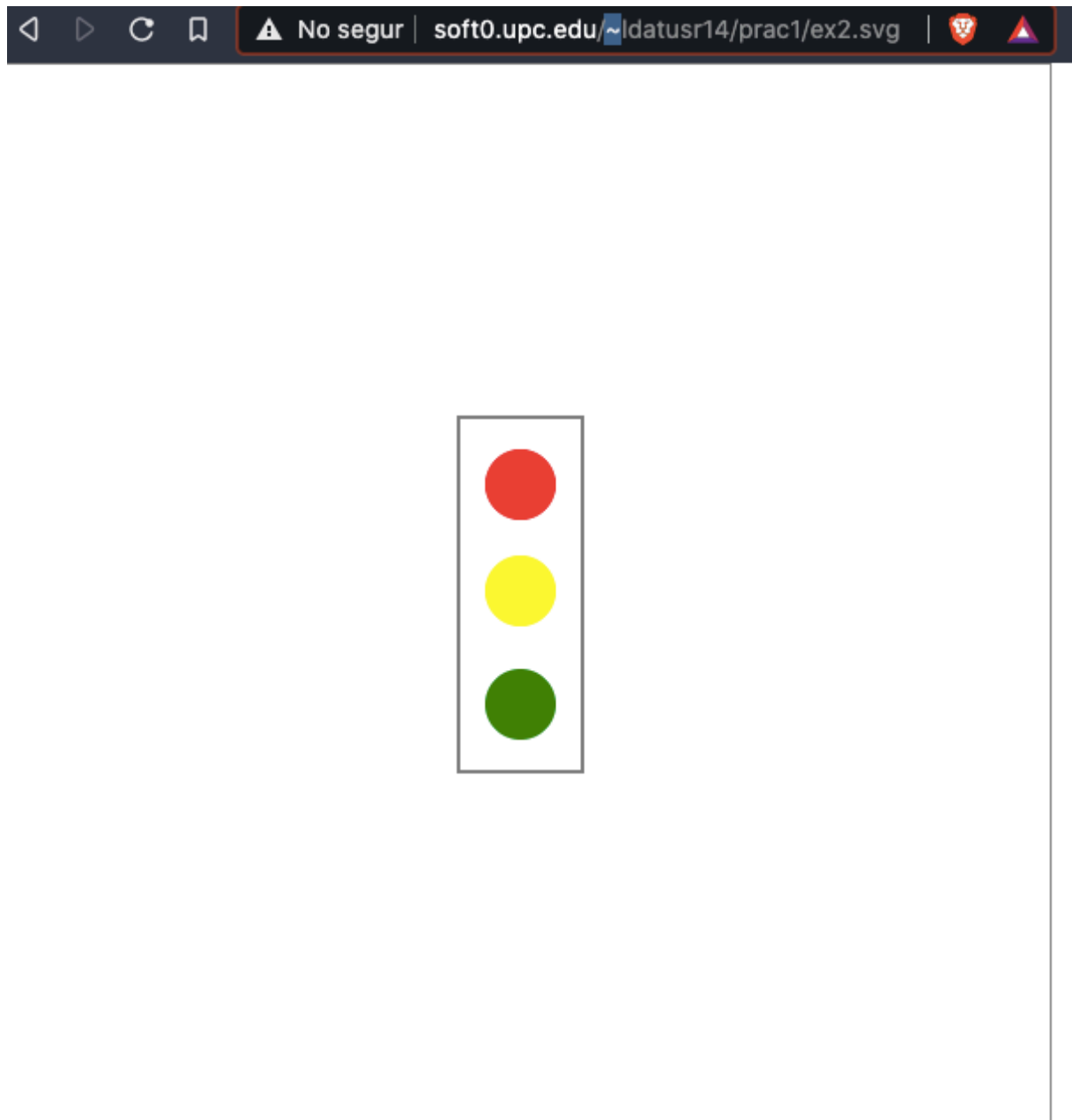
colorCircle c y = colored c (translated 0 y (solidCircle 1))

frame = rectangle 2.5 7.5

trafficLight = frame <> colorCircle green (-2.5) <> colorCircle
yellow 0 <> colorCircle red 2.5
```

```
main :: IO ()  
main = svgOf myDrawing
```

Si el compilem i mirem l'output que ens dona:



Exercici 3

He anat modificant el codi del fitxer `myDrawing.hs` en cada exercici, per això és possible que només veiem un fitxer per a tots els exercicis de la primera part.

Codi:

```
import Drawing

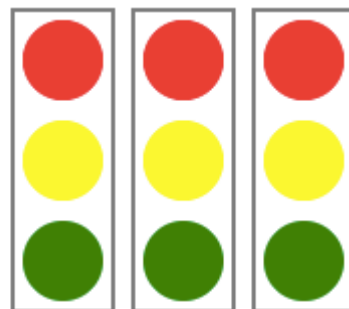
colorCircle c y = colored c (translated 0 y (solidCircle 1))
frame = rectangle 2.5 7.5
trafficLight = frame <> colorCircle green (-2.5) <> colorCircle
yellow 0 <> colorCircle red 2.5

lights :: Int -> Drawing
lights 0 = blank
lights n = translated (3 * fromIntegral n) 0 trafficLight <>
lights (n-1)

myDrawing = lights 3

main :: IO ()
main = svgOf myDrawing
```

Output:



Ara que tenim una fila de 3, anem a fer la columna també de 3:

```
import Drawing

colorCircle c y = colored c (translated 0 y (solidCircle 1))
frame = rectangle 2.5 7.5
trafficLight = frame <> colorCircle green (-2.5) <> colorCircle
yellow 0 <> colorCircle red 2.5

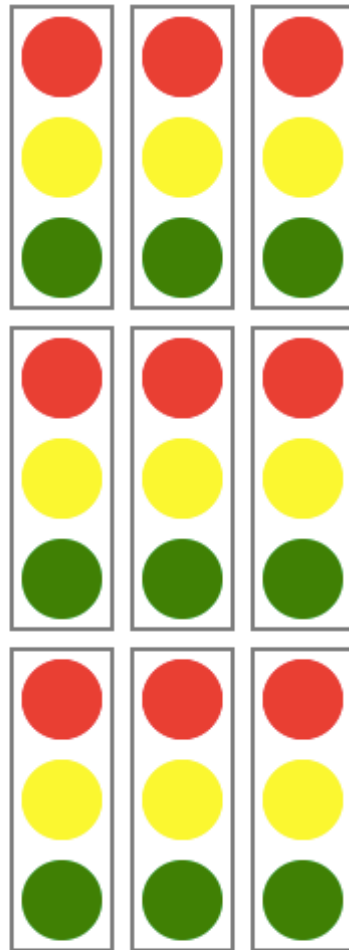
lights :: Int -> Drawing
```

```
lights (-2) = blank
lights n = translated (3 * fromIntegral n) 0 trafficLight <>
lights (n-1)
row y = (translated 0 (y) (lights 1))
matrix = row 8 <> row 0 <> row (-8)

myDrawing = matrix

main :: IO ()
main = svgOf myDrawing
```

Output:



Exercici 4

Per aquest exercici si que he creat un altre fitxer ja que és diferent als dels semàfors.

Codi:

```
import Drawing

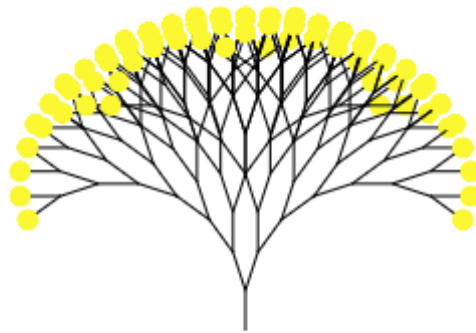
trunk :: Drawing
trunk = polyline [(0,0),(0,1)]
```

```
tree :: Int -> Drawing
tree 0 = colored yellow (circle 0.25)
tree m = trunk <> translated 0 1 (rotated(pi/10) branch) <>
translated 0 1 (rotated(-pi/10) branch)
    where branch = tree (m-1)

myDrawing = tree 8
main :: IO ()
main = svgOf myDrawing
```

Afegeixo directament la foto amb les flors, sense només caldria posar $m = 0$.

Output:



Exercici 5

Fins ara hem estat fent repeticions d'un cert patró nosaltres mateixos, ara ho farem amb una funció `repeatDraw`, tot i que Haskell incorpora una funció que també resol aquest problema (`foldMap :: (Foldable t, Monoid m) => (a -> m) -> t a -> m`)

codi:

```
import Drawing
```

```

colorCircle c y = colored c (translated 0 y (solidCircle 1))
frame = rectangle 2.5 7.5
trafficLight = frame <> colorCircle green (-2.5) <> colorCircle
yellow 0 <> colorCircle red 2.5

```

```

repeatDraw :: (Int -> Drawing) -> Int -> Drawing
repeatDraw thing 0 = blank
repeatDraw thing n = thing n <> repeatDraw thing (n-1)

```

```

myDrawing = repeatDraw lightRow 3
lightRow :: Int -> Drawing
lightRow r = repeatDraw (light r) 3
light :: Int -> Int -> Drawing
light r c = translated (3 * fromIntegral c - 6) (8*fromIntegral
r-16) trafficLight

```

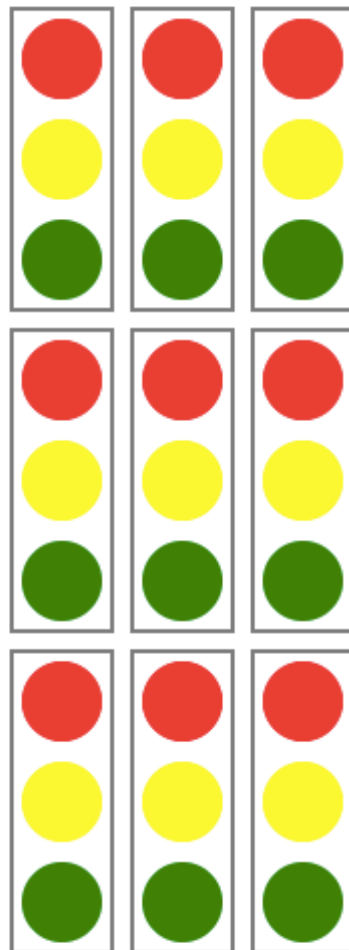
```

main :: IO ()
main = svgOf myDrawing

```

Executem `run-main myDrawing.hs > ~/public_html_/prac1/ex5.svg`

Output



Exercici 6

Donada una llista de n punts, dibuixar n semàfors situats en les corresponents posicions, usant les funcions *foldMap* i *trafficLight*.

codi

```
import Drawing
```

```
colorCircle c y = colored c (translated 0 y (solidCircle 1))  
frame = rectangle 2.5 7.5
```

```

trafficLight = frame <> colorCircle green (-2.5) <> colorCircle
yellow 0 <> colorCircle red 2.5

trafficLights :: [(Double, Double)] -> Drawing
light :: (Double, Double) -> Drawing
light r = translated (fst r) (snd r) trafficLight
trafficLights list = foldMap light list

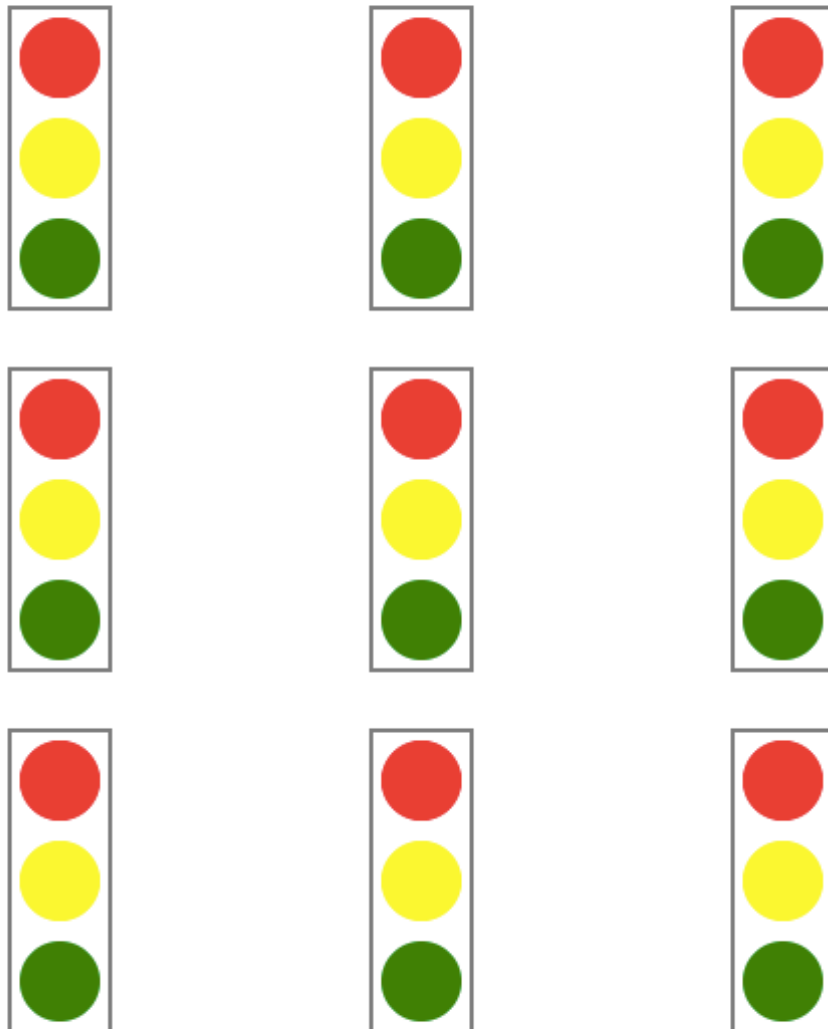
cord = [(0,0), (0,9), (0,-9), (9, 0), (-9, 0), (9,9), (-9,9), (9,
-9), (-9, -9)]

myDrawing :: Drawing
myDrawing = trafficLights cord

main :: IO ()
main = svgOf myDrawing

```

Output



Segona Part: Interacció i modificació d'estat

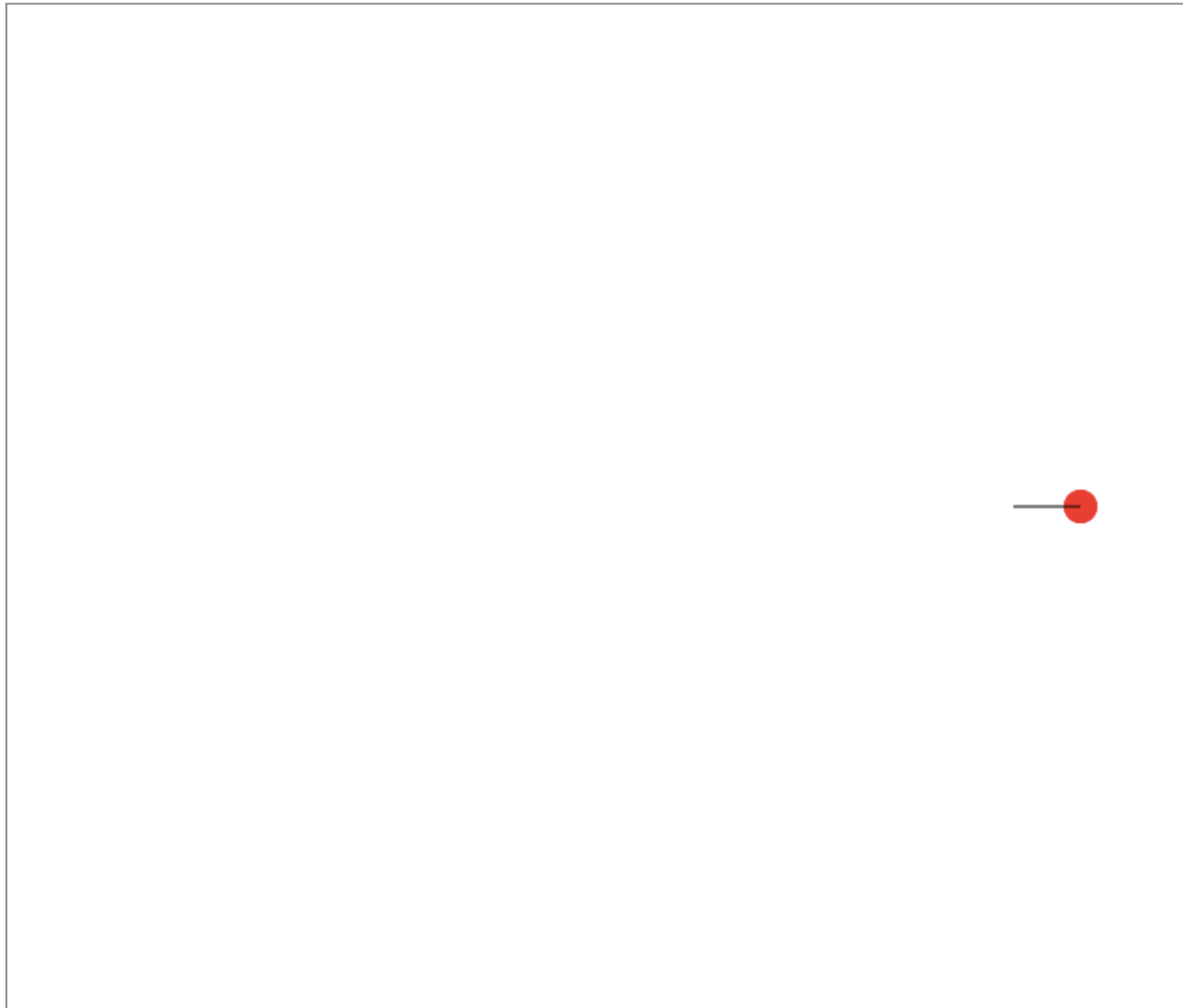
En aquesta part veurem el joc de la vida basat en la màquina de Turing d'Alan Turing.

Per fer-ho primer descarreguem el fitxer prog-2-fun.zip i l'enviem al nostre usuari de DAT, on el descomprimim.

Llavors exectuem `bin/make-cgi src/exemple.hs`, el qual ens compila els arxius que utilitzarem per aquesta part. En el meu cas he hagut de crear el directori *practica1* a dins de *publichtml* per a que funcione tot bé.

Observem al navegador la següent URL: <http://soft0.upc.edu/ldatusr14/practica1/exemple.cgi>

/~ldatusr14/practica1/exemple.cgi



Time: 17.356

Pending events: 0

Last event: KeyUp "META"

Refreshs by second: 10

Pending RCV: WAITING

Exercici 1

En aquesta primera part simplement creem el taulell del joc.

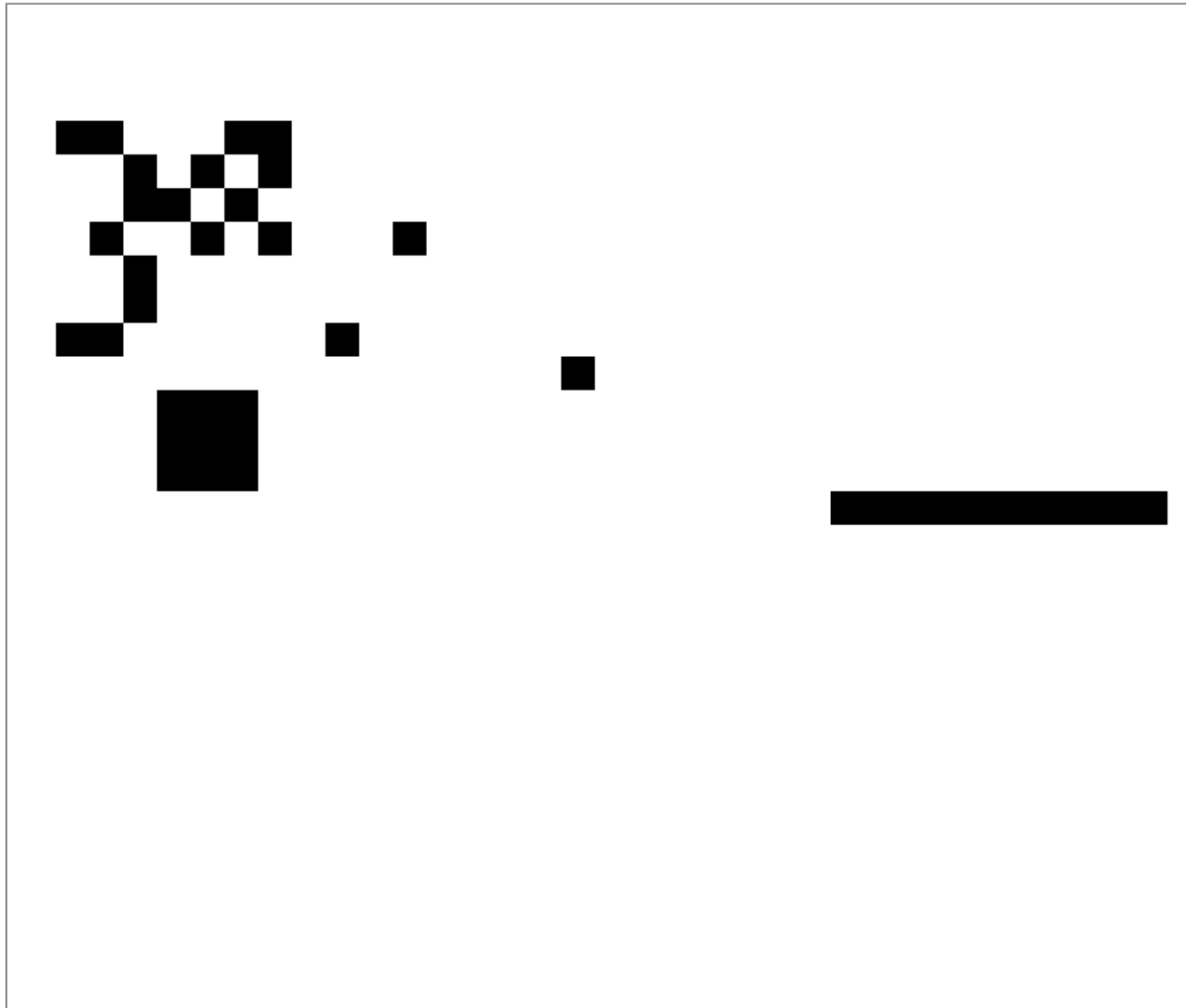
Editem l'arxiu `Draw` del directori `Life` tal que:

```
Draw.hs x life-1.hs life-2.hs
prog-fun-2 > src > Life > Draw.hs
1
2 module Life.Draw where
3 import Life.Board
4 import Drawing
5
6 drawBoard :: Board -> Drawing
7 drawBoard board = foldMap (drawCell board) (liveCells board)
8
9 drawCell :: Board -> Pos -> Drawing
10 drawCell board (x,y) = translated (fromIntegral x) (fromIntegral y) (solidRectangle 1 1)
11
12 -- 'drawGrid minPos maxPos' obte el dibuix d'una graella que inclou els 2 extrems indicats.
13 -- 'minPos' es la posicio de l'extrem (esquerra, inferior) i
14 -- 'maxPos' es la posicio de l'extrem (dreta, superior)'.
15 -- NOTA: Aquesta funcio s'usa a partir del segon pas de la practica.
16 drawGrid :: Pos -> Pos -> Drawing
17 drawGrid minPos maxPos =
18   colored gray $
19     foldMap vline [fst minPos .. fst maxPos + 1]
20     <-> foldMap hline [snd minPos .. snd maxPos + 1]
21   where
22     hline row = polyline [(intToCoord (fst minPos), intToCoord row), (intToCoord (fst maxPos + 1), intToCoord row)]
23     vline col = polyline [(intToCoord col, intToCoord (snd minPos)), (intToCoord col, intToCoord (snd maxPos + 1))]
24     intToCoord i = fromIntegral i - 0.5
25
26
27
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL 1: bash

```
pract1 practical
ldatusr14@soft00:~/public_html$ cd ..
ldatusr14@soft00:~$ cd prog-fun-2/
ldatusr14@soft00:~/prog-fun-2$ ls
bin build src
ldatusr14@soft00:~/prog-fun-2$ bin/make-cgi src/life-1.hs
[1 of 3] Compiling Life.Board      ( /home/pract/LabDAT/ldatusr14/prog-fun-2/src/Life/Board.hs, /home/pract/LabDAT/ldatusr14/prog-fun-2/build/Life/Board.o )
[2 of 3] Compiling Life.Draw      ( /home/pract/LabDAT/ldatusr14/prog-fun-2/src/Life/Draw.hs, /home/pract/LabDAT/ldatusr14/prog-fun-2/build/Life/Draw.o )
[3 of 3] Compiling Main          ( src/life-1.hs, /home/pract/LabDAT/ldatusr14/prog-fun-2/build/Main.o )
Linking /home/pract/LabDAT/ldatusr14/public_html/practical/life-1.cgi ...
S'ha instal·lat el CGI "/home/pract/LabDAT/ldatusr14/public_html/practical/life-1.cgi".
Aquest CGI serà accessible des d'un navegador amb la URL "http://soft0.upc.edu/~ldatusr14/practical/life-1.cgi"
ldatusr14@soft00:~/prog-fun-2$
```


/~ldatusr14/practica1/life-1.cgi



Time: 122.567

Pending events: 0

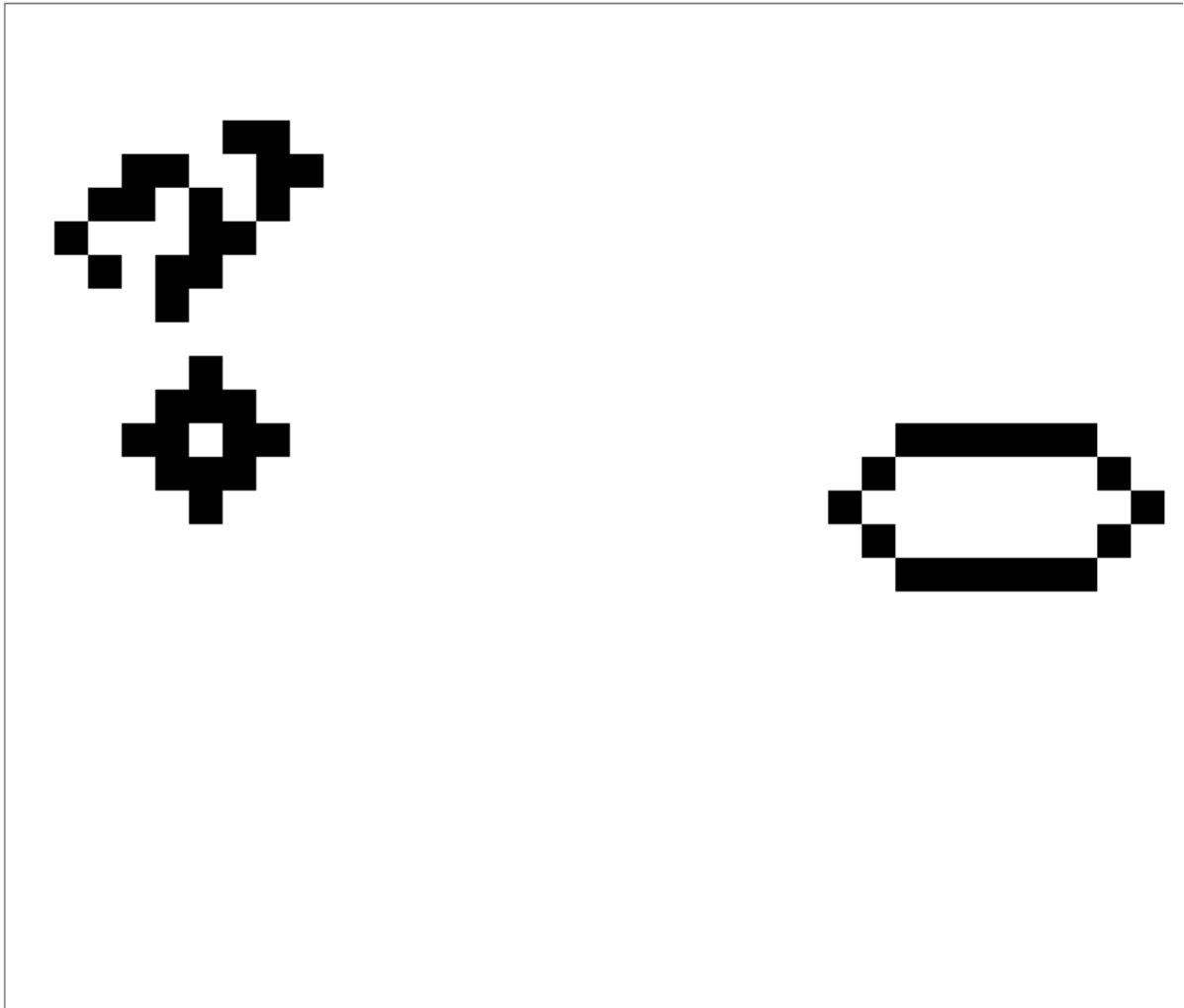
Last event: KeyUp "META"

Refreshs by second:  15

Pending RCV: WAITING

Veiem com podem anar apretant celes i aquestes es tornen de color negre (vives) o tornar-les a apretar perquè estiguin blanques (mortes), i que si apremem la tecla 'N' passem a la següent generació.

/~ldatusr14/practica1/life-1.cgi



Time: 447.432

Pending events: 0

Last event: KeyUp "META"

Refreshs by second: 15

Pending RCV: WAITING

Exercici 2

Al segon pas consisteix en afegir un control per mostrar i ocultar una graella en el taulell , així és més fàcil visualitzar l'espai per les diferents cel·les.

Per fer-ho completem el mòdul `Main` (`src/life-2.hs`)

```
38
39 initial = Game
40 { gmBoard = foldr (setCell True) initBoard board0Cells
41   , gmGridMode = NoGrid
42 }
43
44 -----
45 -- A completar per l'estudiant
46
47
48 handleEvent (KeyDown "G") game =
49   case (gmGridMode game) of
50     NoGrid -> setGmGridMode LivesGrid game
51     LivesGrid -> setGmGridMode ViewGrid game
52     ViewGrid -> setGmGridMode NoGrid game
53
54 handleEvent _ game =
55   game
56
57
58 draw game =
59   let minCorner = (round((viewWidth/2)*(-1)) , round((viewHeight/2)*(-1)))
60       maxCorner = (round(viewWidth/2) , round(viewHeight/2))
61   in
62     case (gmGridMode game) of
63       NoGrid -> drawBoard (gmBoard game)
64       LivesGrid -> drawBoard (gmBoard game) <> drawGrid (minLiveCell (gmBoard game)) (maxLiveCell (gmBoard game))
65       ViewGrid -> drawBoard (gmBoard game) <> drawGrid (minCorner) (maxCorner)
66
67
```


PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL 1: bash

```
ldatusr14@soft00:~/prog-fun-2/src$ cd ..
ldatusr14@soft00:~/prog-fun-2$ bin/make-cgi src/life-2.hs
[3 of 3] Compiling Main ( src/life-2.hs, /home/pract/LabDAT/ldatusr14/prog-fun-2/build/Main.o )
src/life-2.hs:60:43:
  A section must be enclosed in parentheses
  thus: (round (viewHeight / 2) *)
ldatusr14@soft00:~/prog-fun-2$ bin/make-cgi src/life-2.hs
[3 of 3] Compiling Main ( src/life-2.hs, /home/pract/LabDAT/ldatusr14/prog-fun-2/build/Main.o )
Linking /home/pract/LabDAT/ldatusr14/public_html/practica1/life-2.cgi ...
S'ha instalat el CGI '/home/pract/LabDAT/ldatusr14/public_html/practica1/life-2.cgi'.
Aquest CGI serà accessible des d'un navegador amb la URL 'http://soft0.upc.edu/~ldatusr14/practica1/life-2.cgi'
ldatusr14@soft00:~/prog-fun-2$
```

Veiem l'output a la URL <http://soft0.upc.edu/ldatusr14/practica1/life-2.cgi>

Browser address bar: `soft0.upc.edu/dat/drawing-0.2/cgi-static/index.html?appCgi=%2F~ldatusr14%2Fpractica1%2Flife-2.cgi`

Page title: `/~ldatusr14/practica1/life-2.cgi`



Time: 35.139
Pending events: 0
Last event: KeyUp
"META"


Refreshes by second: 10
Pending RCV: READY

La velocitat de refresc està limitada pel RTT (round-trip time) entre el vostre navegador i el servidor. Per exemple, si el RTT de la xarxa entre el navegador i el servidor és de 500ms no es podrà obtenir una velocitat de refresc superior a 2 per segon.

Per canviar de mode premem la tecla "G":

Browser address bar: `soft0.upc.edu/dat/drawing-0.2/cgi-static/index.html?appCgi=%2F~ldatusr14%2Fpractica1%2Flife-2.cgi`

Page title: `/~ldatusr14/practica1/life-2.cgi`

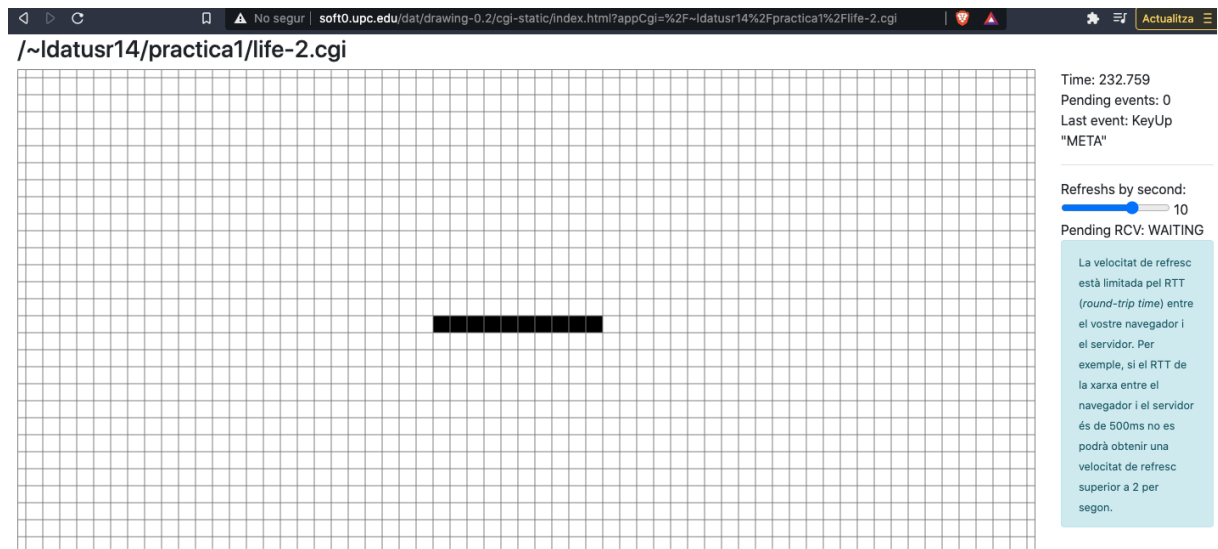


Time: 223.76
Pending events: 0
Last event: KeyUp
"META"

Refreshes by second: 10
Pending RCV: READY

La velocitat de refresc està limitada pel RTT (round-trip time) entre el vostre navegador i el servidor. Per exemple, si el RTT de la xarxa entre el navegador i el servidor és de 500ms no es podrà obtenir una velocitat de refresc superior a 2 per segon.

Si la premem un altre cop passem al ViewGridMode



Exercici 3

En el tercer pas introduïrem controls per poder fer zoom i desplaçaments en la visualització del taulell.

Per això, definim nous camps `gmZoom` i `gmShift` en l'estat del joc `Game` al fitxer `src/life-3.hs` del projecte.

Veiem el codi que hem desenvolupat per aquest exercici:

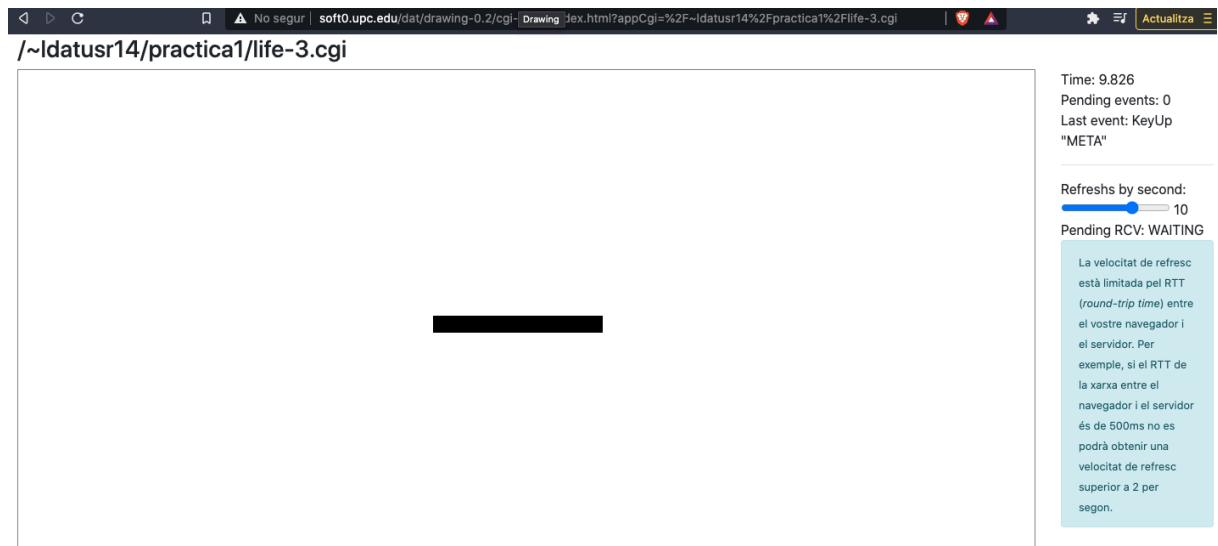
```

life-2.hs  Draw.hs  life-4.hs  life-3.hs  X
prog-fun-2 > src > life-3.hs
48
49 -----
50 -- A completar per l'estudiant
51
52 handleEvent (KeyDown "G") game =
53   case (gmGridMode game) of
54     NoGrid -> setGmGridMode LivesGrid game
55     LivesGrid -> setGmGridMode ViewGrid game
56     ViewGrid -> setGmGridMode NoGrid game
57
58 handleEvent (KeyDown "I") game =
59   if gmZoom game < 2.0 then setGmZoom (gmZoom game * 2.0) game
60   else game
61
62 handleEvent (KeyDown "O") game =
63   if gmZoom game < 5.0 then setGmZoom (gmZoom game / 2.0) game
64   else game
65
66 handleEvent (KeyDown "ARROWUP") game = setGmShift (gmShift game ^~ (1.0 / gmZoom game) *^(0,5)) game
67
68 handleEvent (KeyDown "ARROWDOWN") game = setGmShift (gmShift game ^~ (1.0 / gmZoom game) *^(0,-5)) game
69
70 handleEvent (KeyDown "ARROWRIGHT") game = setGmShift (gmShift game ^~ (1.0 / gmZoom game) *^(5,0)) game
71
72 handleEvent (KeyDown "ARROWLEFT") game = setGmShift (gmShift game ^~ (1.0 / gmZoom game) *^(-5,0)) game
73
74 handleEvent _ game =
75   game
76
77 pointToPos :: Point -> Game -> Pos
78 pointToPos p game =
79   let (gx, gy) = (1.0 / gmZoom game) *^ p ^~ gmShift game
80   in ( round gx, round gy )
81
82 draw game =
83   let (x,y) = gmShift game
84       minCorner = (round((viewWidth/2)*(-1)) , round((viewHeight/2)*(-1)))
85       maxCorner = (round((viewWidth/2)) , round((viewHeight/2)))
86   in scaled (gmZoom game) (gmZoom game) $ translated x y (drawBoard (gmBoard game)) <>
87   case gmGridMode game of
88     NoGrid -> blank
89     LivesGrid -> drawGrid (minLiveCell (gmBoard game)) (maxLiveCell (gmBoard game))
90     ViewGrid -> drawGrid minCorner maxCorner
91

```

Per veure l'output anem a la URL <http://soft0.upc.edu/ldatusr14/practica1/life-3.cgi>

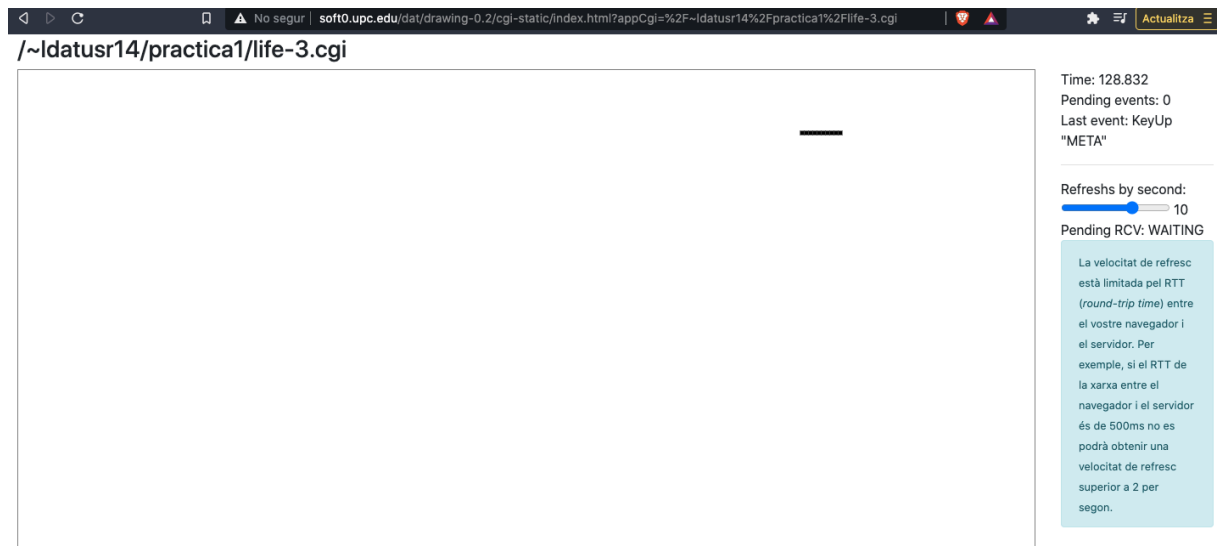
Estat inicial:



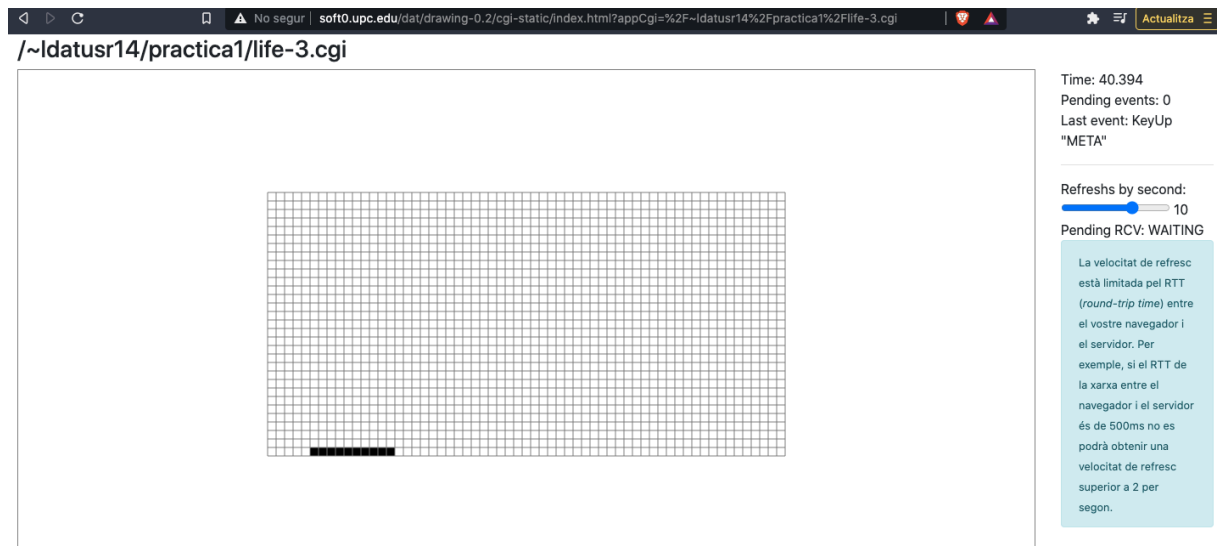
Zoom In (apretant tecla "I"):



Apretant tecla "O" (Zoom Out) i algunes ARROWRIGHT i ARROWDOWN:



Utilitzant el "GridMode" i movent-nos cap a l'esquerra i avall desde l'estat inicial:



He vist que si utilitzes el LivesGrid mode (i també amb el ViewGrid) la graella no s'actualitza quan es mou pel taulell, però qua reinicies a l'estat inicial ho pots executar correctament.

Exercici 4

En el quart pas introduïrem el temps, evolució automàtica de generacions, i controls per poder canviar el mode de pausa/evolució automàtica i la velocitat de l'evolució automàtica.

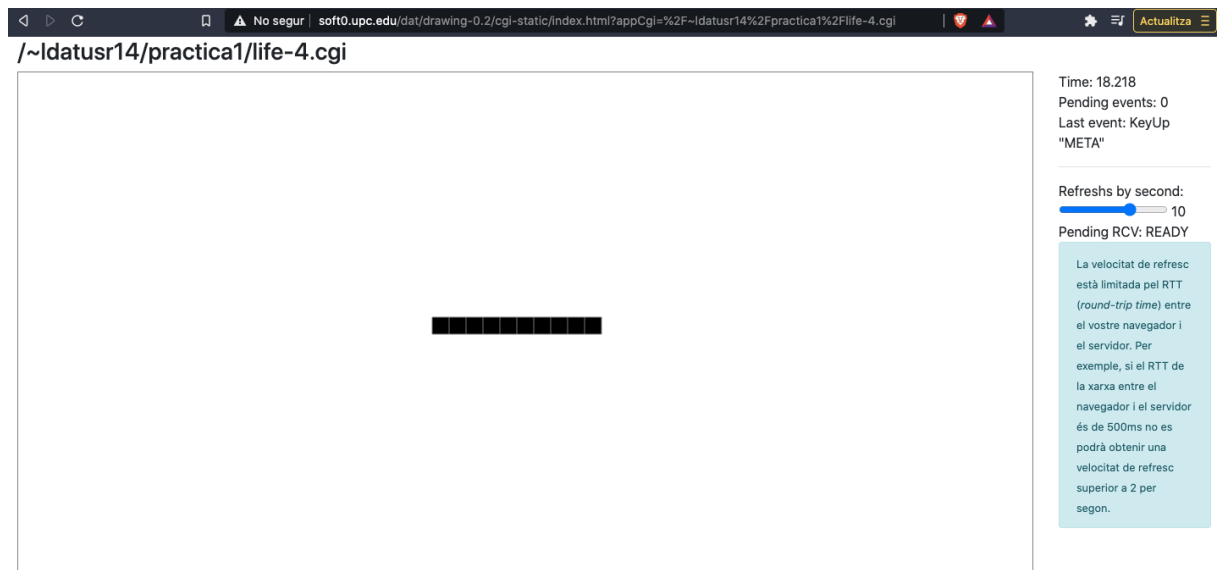
El codi per aquest exercici és el següent:


```

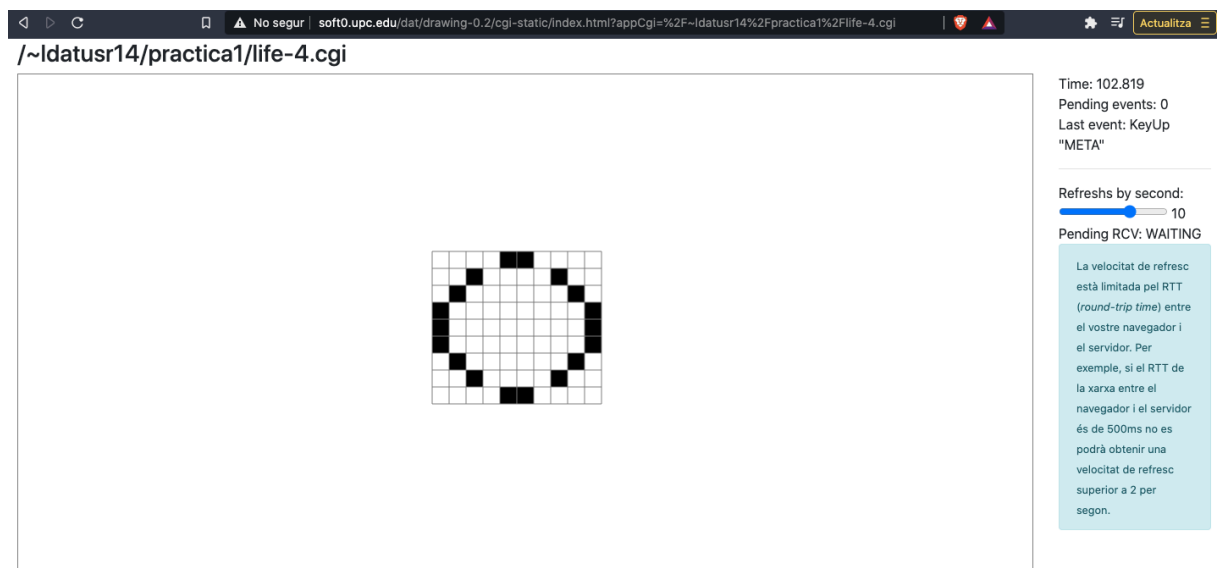
prog-fun-2 > src > ✖ life-4.hs
56 | }
57 |
58 | -----
59 | -- A completar per l'estudiant
60 |
61 | handleEvent (KeyDown "G") game =
62 |   case (gmGridMode game) of
63 |     NoGrid -> setGmGridMode LivesGrid game
64 |     LivesGrid -> setGmGridMode ViewGrid game
65 |     ViewGrid -> setGmGridMode NoGrid game
66 |
67 | handleEvent (KeyDown "I") game =
68 |   if gmZoom game < 2.0 then setGmZoom (gmZoom game * 2.0) game
69 |   else game
70 |
71 | handleEvent (KeyDown "O") game =
72 |   if gmZoom game < 5.0 then setGmZoom (gmZoom game / 2.0) game
73 |   else game
74 |
75 | handleEvent (KeyDown "ARROWUP") game = setGmShift (gmShift game ^~^ (1.0 / gmZoom game) *^(0,5)) game
76 |
77 | handleEvent (KeyDown "ARROWDOWN") game = setGmShift (gmShift game ^~^ (1.0 / gmZoom game) *^(0,-5)) game
78 |
79 | handleEvent (KeyDown "ARROWRIGHT") game = setGmShift (gmShift game ^~^ (1.0 / gmZoom game) *^(5,0)) game
80 |
81 | handleEvent (KeyDown "ARROWLEFT") game = setGmShift (gmShift game ^~^ (1.0 / gmZoom game) *^(-5,0)) game
82 |
83 | handleEvent (KeyDown " ") game =
84 |   setGmPaused (not $ gmPaused game) game
85 |
86 | handleEvent (KeyDown "+") game =
87 |   if (gmInterval game / 2) >= 0.125 then setGmInterval (gmInterval game / 2) game
88 |   else game
89 |
90 | handleEvent (KeyDown "-") game =
91 |   setGmInterval (gmInterval game * 2) game
92 |
93 | handleEvent (TimePassing dt) game =
94 |   if gmPaused game == False then
95 |     (if (gmElapsedTime game + dt >= gmInterval game)
96 |      then setGmElapsedTime 0 (setGmBoard (nextGeneration $ gmBoard game) game)
97 |      else setGmElapsedTime (gmElapsedTime game + dt) game)
98 |   else game
99 |
100 | handleEvent (MouseDown (x,y)) game =
101 |   let pos = pointToPos (x,y) game
102 |   | brd = gmBoard game
103 |   in setGmBoard (setCell (not $ cellIsLive pos brd) pos brd) game
104 |
105 | handleEvent _ game =
106 |   game
107 |
108 | pointToPos :: Point -> Game -> Pos
109 | pointToPos p game =
110 |   let (gx, gy) = (1.0 / gmZoom game) *^ p ^~^ gmShift game
111 |   in ( round gx, round gy )
112 |
113 | draw game =
114 |   let (x,y) = gmShift game
115 |   | minCorner = (round((viewWidth/2)*(-1)) , round((viewHeight/2)*(-1)))
116 |   | maxCorner = (round((viewWidth/2)) , round((viewHeight/2)))
117 |   in scaled (gmZoom game) (gmZoom game) $ translated x y (drawBoard (gmBoard game)) <>
118 |   case gmGridMode game of
119 |     NoGrid -> blank
120 |     LivesGrid -> drawGrid (minLiveCell (gmBoard game)) (maxLiveCell (gmBoard game))
121 |     ViewGrid -> drawGrid minCorner maxCorner
122 |
123 |

```

Output estat inicial amb grid activat:




Si apremem l'espai " " veiem com evolucionen les cel·les a la següent generació, i podem regular el temps que volem que canviïn de generació amb les teclet "+" i "-".



soft0.upc.edu/dat/drawing-0.2/cgi-static/index.html?appCgi=%2F~ldatusr14%2Fpractica1%2Flife-4.cgi

/~ldatusr14/practica1/life-4.cgi



Time: 109.92
Pending events: 0
Last event: KeyUp
"META"

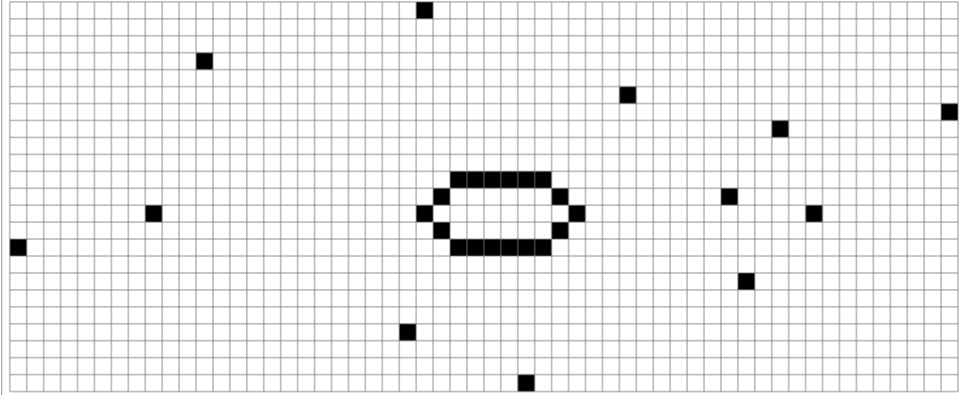
Refreshes by second: 10
Pending RCV: WAITING

La velocitat de refresc està limitada pel RTT (round-trip time) entre el vostre navegador i el servidor. Per exemple, si el RTT de la xarxa entre el navegador i el servidor és de 500ms no es podrà obtenir una velocitat de refresc superior a 2 per segon.

També les podem moure com a l'anterior exercici pel taulell, i activar noves cel·les, les quals també canviaran de generació si apremem l'espai.

soft0.upc.edu/dat/drawing-0.2/cgi-static/index.html?appCgi=%2F~ldatusr14%2Fpractica1%2Flife-4.cgi

/~ldatusr14/practica1/life-4.cgi



Time: 200.627
Pending events: 0
Last event: KeyUp
"META"

Refreshes by second: 10
Pending RCV: READY

La velocitat de refresc està limitada pel RTT (round-trip time) entre el vostre navegador i el servidor. Per exemple, si el RTT de la xarxa entre el navegador i el servidor és de 500ms no es podrà obtenir una velocitat de refresc superior a 2 per segon.

No segur
soft0.upc.edu/dat/drawing-0.2/cgi-static/index.html?appCgi=%2F~ldatusr14%2Fpractica1%2Flife-4.cgi

/~ldatusr14/practica1/life-4.cgi

Time: 340.737
Pending events: 0
Last event: KeyUp
"META"

Refreshes by second:

Pending RCV: WAITING

La velocitat de refresc està limitada pel RTT (round-trip time) entre el vostre navegador i el servidor. Per exemple, si el RTT de la xarxa entre el navegador i el servidor és de 500ms no es podrà obtenir una velocitat de refresc superior a 2 per segon.

Exercici 5

En aquest últim exercici afegirem en el dibuix una mica de text que expliqui l'estat d'algun paràmetre del joc, i en mode pausa, una ajuda de les tecles que canvien els paràmetres del joc.

Per fer-ho modifiquem el codi de Main de l'apartat anterior tal que:

```

... life-2.hs Draw.hs life-4.hs x
prog-run-2 > src > life-4.hs
>0 { gmBoard = foldr (setCell true) initBoard boardCells
51   , gmGridMode = NoGrid
52   , gmZoom = 1.0, gmShift = (0.0, 0.0)
53   , gmPaused = true
54   , gmInterval = 1.0 -- in seconds
55   , gmElapsedTime = 0.0
56 }
57
58
59 -- A completar per l'estudiant
60
61 keysNormes = ["N", "G", "O", "I", "ARROWUP", "ARROWDOWN", "ARROWRIGHT", "ARROWLEFT", "SPACE", "+", "-", "Click with your mouse to set live/dead cells"]
62 valueNormes = ["Next Step", "Change to grid mode", "Zoom out", "Zoom in", "Shift Down", "Shift Up", "Shift Left", "Shift Right", "Toggle pause/run", "Increase run velocity", "Decrease run velocity"]
63
64 funcAux :: Game -> [String]
65 funcAux g = [show (1.0/gmInterval g) ++ " steps per second" ++ (if (gmPaused g) then " (paused)" else ""), "Zoom = " ++ show(gmZoom g), "Shift = " ++ show (fst(gmShift g)) ++ ", " ++ show (snd(gmShift g))]
66
67 newLine :: [String] -> String -> Drawing
68 newLine [] a = blank
69 newLine (l:ls) a = translated 0 (-1) ((latex a l) <- (newLine ls a))
70
71 handleEvent (KeyDown "C") game =
72   case (gmGridMode game) of
73     NoGrid -> setGmGridMode LivesGrid game
74     LivesGrid -> setGmGridMode ViewGrid game

```

```

116 handleEvent _ game =
117   game
118
119 pointToPos :: Point -> Game -> Pos
120 pointToPos p game =
121   let (gx, gy) = (1.0 / gmZoom game) *^ p ^^^ gmShift game
122   in ( round gx, round gy )
123
124 draw game =
125   let (x,y) = gmShift game
126       minCorner = (round((viewWidth/2)*(-1)), round((viewHeight/2)*(-1)))
127       maxCorner = (round((viewWidth/2)), round((viewHeight/2)))
128   in scaled (gmZoom game) (gmZoom game) $ translated x y (drawBoard (gmBoard game)) <
129   case gmGridMode game of
130     NoGrid -> blank
131     LivesGrid -> drawGrid (minLiveCell (gmBoard game)) (maxLiveCell (gmBoard game))
132     ViewGrid -> drawGrid minCorner maxCorner
133
134   (if gmPaused game then
135     (translated (-viewWidth/2 + 1) (viewHeight/2) (colored blue (newLine keyNormes startAnchor)) <
136     translated (-viewWidth/2 + 6) (viewHeight/2) (colored blue (newLine valueNormes startAnchor)))
137     else blank )
138   <
139   (translated (viewWidth/2 - 1) (viewHeight/2) (colored red (newLine (funcAux game) endAnchor)))
140

```

Veiem el primer output quan entrem a la URL de l'apartat anterior:

The screenshot shows a web browser window with the URL `soft0.upc.edu/ldatusr14/practica1/life-4.cgi`. The page displays a game interface. On the left, there is a legend for controls: 'N Next Step', 'G Change to grid mode', 'O Zoom out', 'I Zoom in', 'ARROWUP Shift Down', 'ARROWDOWN Shift Up', 'ARROWRIGHT Shift Left', 'ARROWLEFT Shift Right', 'SPACE Toggle pause/run', '+ Increase run velocity', and '-' Decrease run velocity. It also says 'Click with your mouse to set live/dead cells'. The main game area is mostly black. On the right side, there is a status panel showing 'Time: 7.183', 'Pending events: 0', 'Last event: KeyUp', and 'META'. Below this, there is a 'Refreshes by second:' control set to 10. A blue box at the bottom right contains text explaining that the refresh rate is limited by the RTT (round-trip time) between the browser and the server, and that a 500ms RTT would prevent a refresh rate higher than 2 per second.

Si apremem espai i ens canviem de generació veiem com desapareix el text d'ajuda, i quan el tornem a apretar aquest torna a aparèixer.

Per veure més, accedir a <http://soft0.upc.edu/ldatusr14/practica1/life-4.cgi>

© Marc Bosch Sarquella, DAT, UPC 2021.