Grids in Code

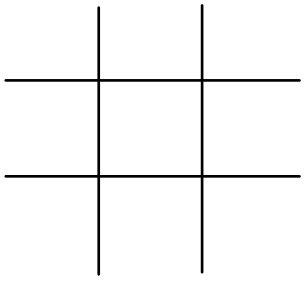
Create an empty grid

A row, maybe the top row?

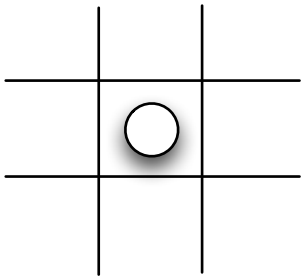
Right most column in top row.

```
>>
>>
<t
>>
>>
>>
[0
>>
```

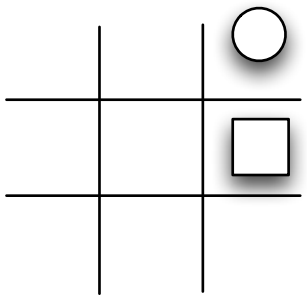
Some Sample Grids



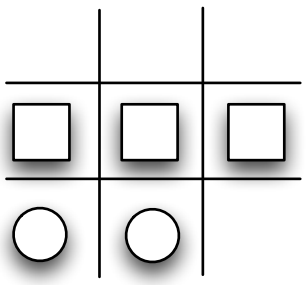
```
[['','',''],
 ['','',''],
 ['','','']]
```



```
·> grid = []
·> type(grid)
:ype 'list'>
·> grid = [0,0,0]
·> grid[2] = "x"
·> grid
), 0, 'x']
·> grid = [[0,0,0], [0,0,0], [0,0,0]]
```



```
[['','','o'],
 ['','','x'],
 ['','','']]
```



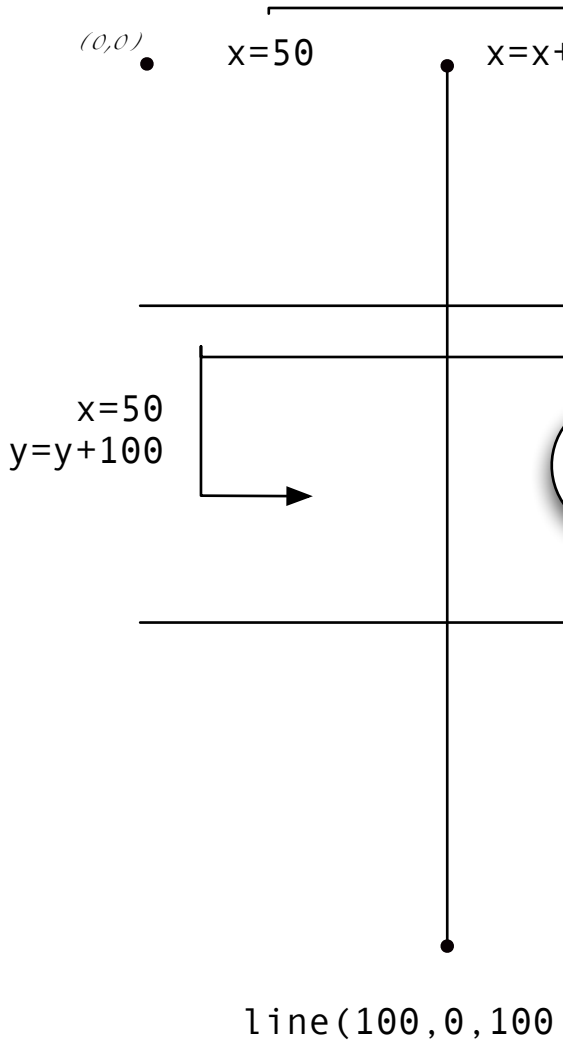
Drawing the Game Board

```
grid = [['','','x']

y = 50
for row in grid:

  x = 50
  for col in row:
    ellipse(x,y,40,4
    x = x + 100

  y = y + 100
```

1) Start at 50,50

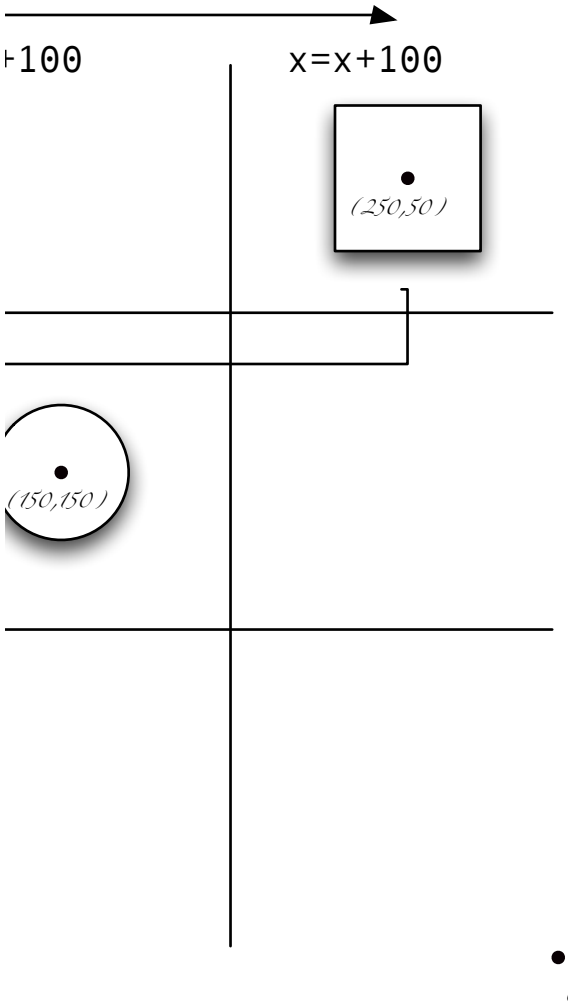
2) Loop through each column in row. Move right 100 each time.

3) Loop each row too (this is the first loop in the code). Move down 100 each time. Reset x back to 50!



, ['','o',''], ['','',''']]

40)



,300)

```
[['', '', ''],
 ['', 'o', ''],
 ['', '', '']]
```

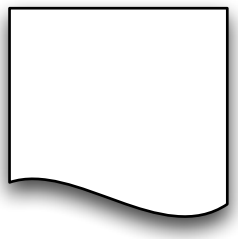
Python and PyProcessing
2D Array vs. Tic Tac Toe

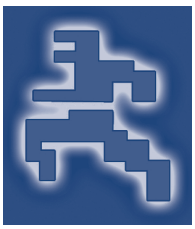
2D arrays (or lists) are powerful beasts. And Tic Tac Toe is a great way to first start using one.

The 2D array can be used in Processing by creating it as a global variables. Then in draw you can use the code sample here to draw the gameboard. In mouseClicked you can find out if the user clicked a positions - update the array and then when draw runs a moment later the game board is updated.

```
[['','',''],
 ['x','x','x'],
 ['o','o','']]
```

Finding Winners





mr.cordiner.com

Hey, if th
same as

```
  Consider using
rectMode(CENTER)
 - see reference
 manual for what
        it does.
```


It is a function of cou

e left one is the same as the centre one and the centre is
the right (and their are not empty, duh) then someone w

Send back who the winner was (and kind of bre

rse.

the

on!

ak).

```
def findWinner():
  for row in grid:
    if row[0] == row[1] and \
    row[1] == row[2] and \
    row[0] != '':
      print "we have a winner!"
      return row[0]
```