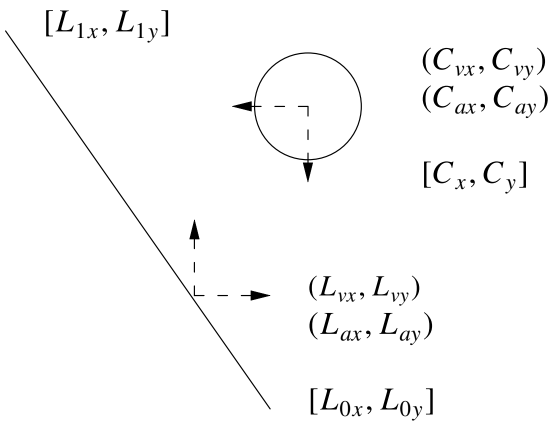
John Romero Programming Proverbs

-

- John Romero, “The Early Days of Id Software - John Romero @ WeAreDevelopers Conference 2017”

Circle line collision

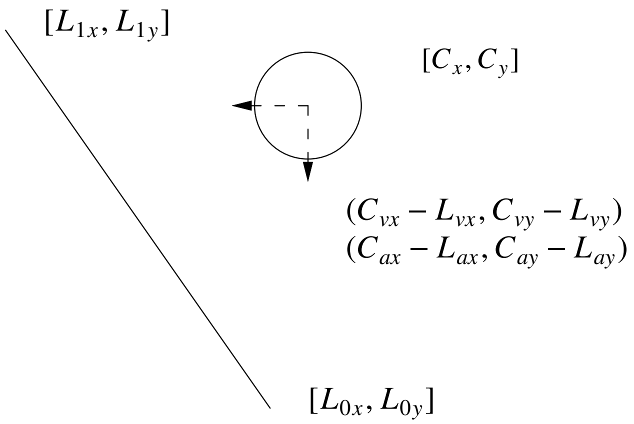
- assumes that we have a robust solution to the circle circle problem which is fully debugged and ready to be used :-)

- 

- at what time is the earliest collision between the line and circle?

Step one

- is to consider the line as stationary and only the circle as moving:



- hence we now have the relative velocity, acceleration between the circle and line
  - the radius of the circle is r

Step two

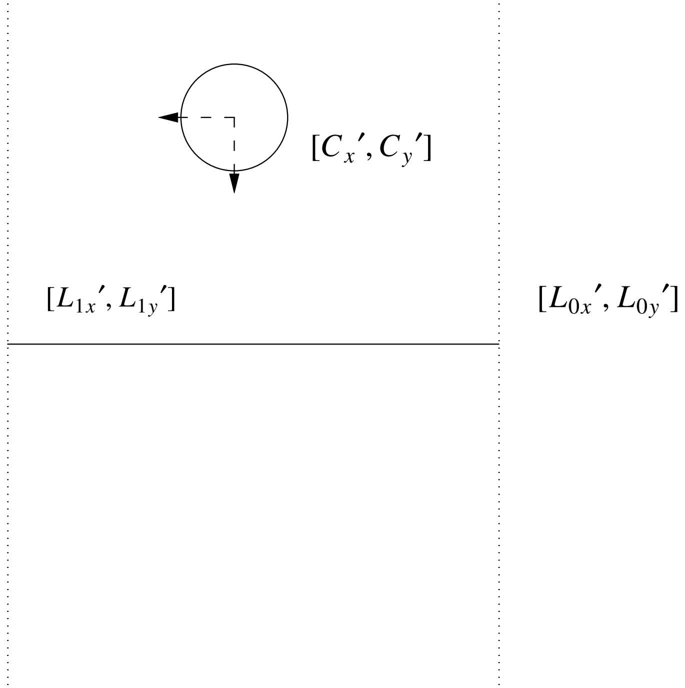
- consider the line to be lying on the X axis, say, with $[L_{1x}, L_{1y}]$ on the point [0, 0]
  - and $[L_{0x}, L_{0y}]$ at [length(L) , 0]

- this involves translating the line and circle by $[-L_{1x}, -L_{1y}]$

- it also involves rotating the line, circle and the relative velocity and acceleration by $\theta = arcsin\left(\frac{L_{0y} - L_{1y}}{\sqrt{(L_{0x} - L_{1x})^2 + (L_{0y} - L_{1y})^2}}\right)$

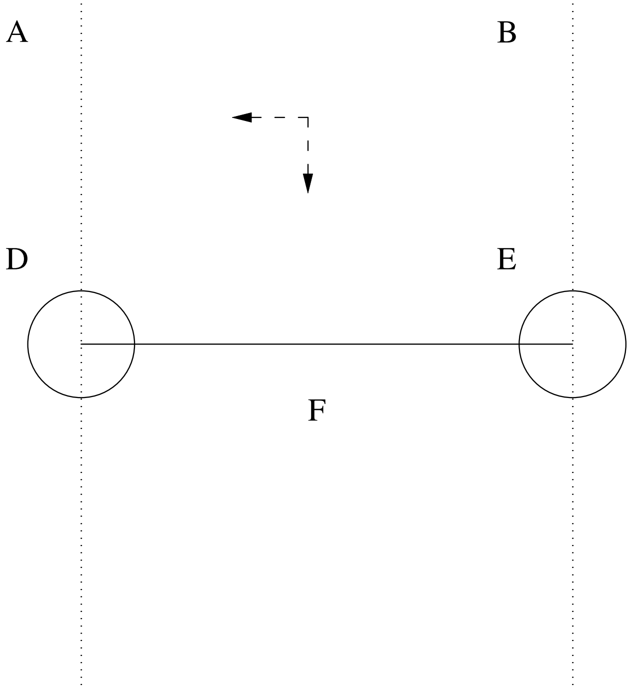
Step two

- we can redraw our diagram as:



Step 3

- redraw the diagram as:



  - the radius of the circle is r

- now we can ask three questions:

Step 3

- (i) does the point C_x', C_y' hit the left circle?
  - in which case the circle will hit the left edge of the line
- (ii) does the point C_x', C_y' hit the right circle?
  - in which case the circle will hit the right edge of the line
- (iii) does the point $C_x', C_y - r'$ hit inbetween the left and right end points of the line?

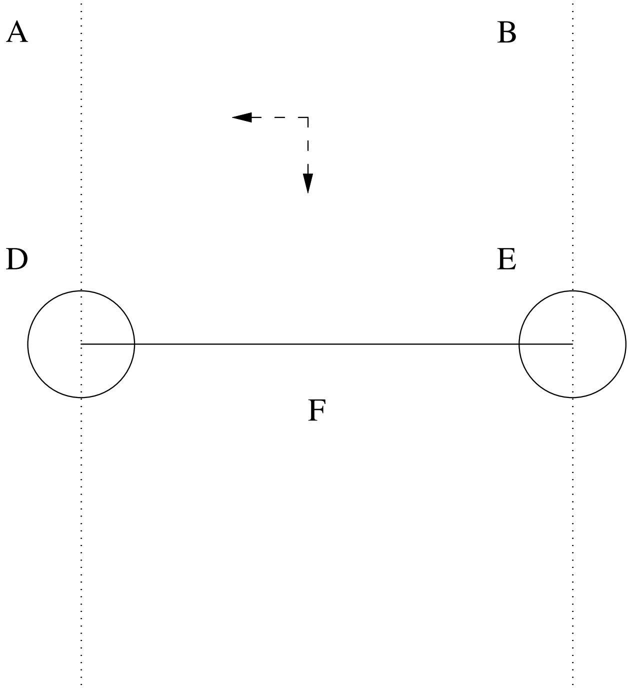
Step 4

- to answer both (i) and (ii) we notice that:

- all we need to do is call our circle circle algorithm and ask this question as whether the new circle hits point (a circle with a radius of 0)

Step 4

- to answer (iii) we return to the diagram:



Step 4

- we need to know at what time, t, our point hits the X axis
- we only need to consider the Y values of the velocity, position, acceleration vectors
- so: $0 = s_0 + ut + \frac{a}{2}t^2$
- and solve for t

Step 4

- now we have a value for t we can find where about on the X axis the point will hit using the same formula, but we plug in the X values of the velocity, position, acceleration vectors

- let s_x be the new position

- if $s_x \geq 0$ and $s_x \leq length(line)$
  - then we have a hit!

- from answering questions (i), (ii), (iii) we ignore any negative time values, only remember the smallest time value >= 0 which tells us the next time of a collision

Calculating the Centre of Gravity of a polygon (centroid of polygon)

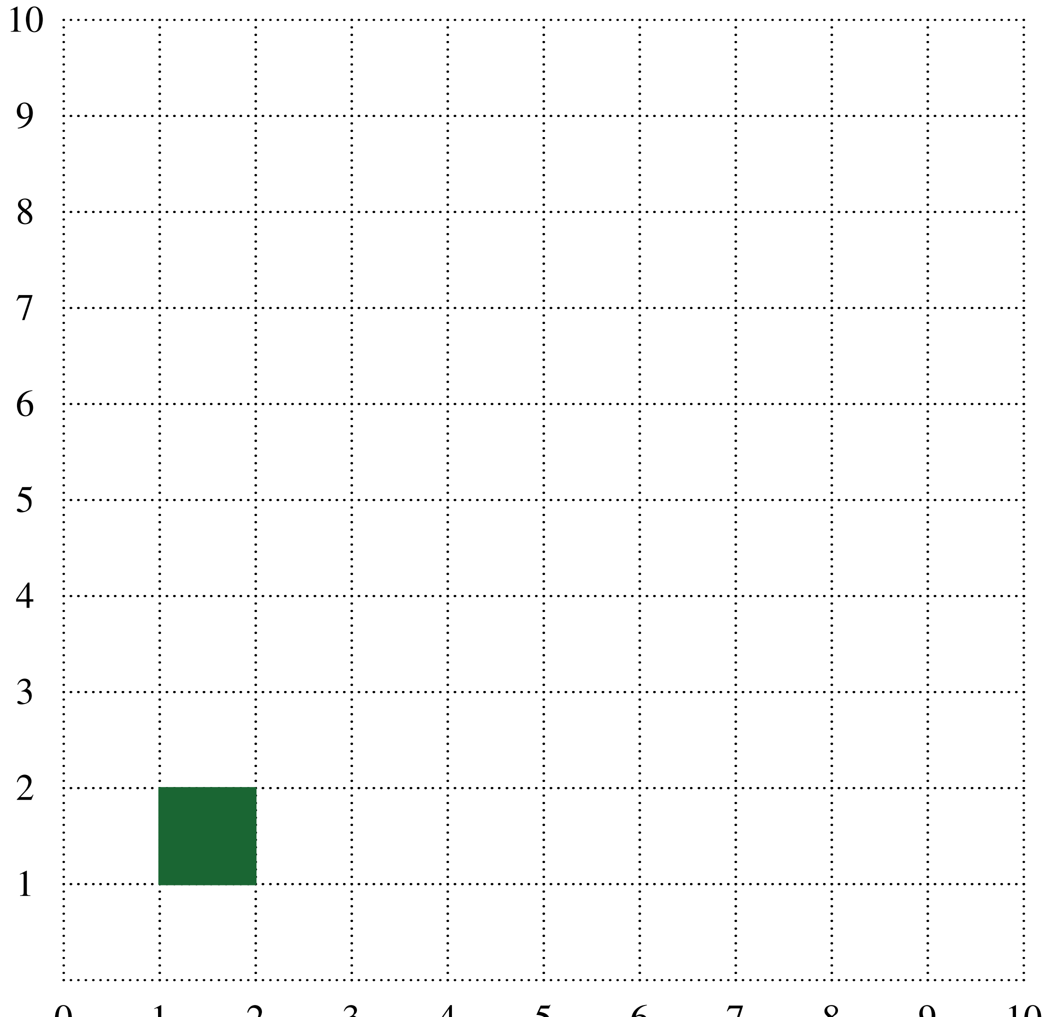
- clearly the centre of gravity for a linear mass circle is its centre
- how do we calculate the centre of gravity for a linear mass polygon?
- centroid of a non-self-intersecting closed polygon defined by a number of points (or vertices):
  - $(x_0, y_{0),}(x_1, y_{1),} \ldots, (x_{n-1}, y_{n-1})$
  - each point or vertice **must** be presented in a clockwise or anticlockwise order
  - we need to find position (c_x, c_y) which is the centre of gravity of this polygon

Calculating the Centre of Gravity of a polygon (centroid of polygon)

- firstly we need to find the polygons area

- $A = \frac{1}{2} \sum_{i=0}^{n-1} x_i y_{i+1} - x_{i+1} y_i$

Calculating the Centre of Gravity of a polygon (centroid of polygon)



Calculating the Centre of Gravity of a polygon (centroid of polygon)

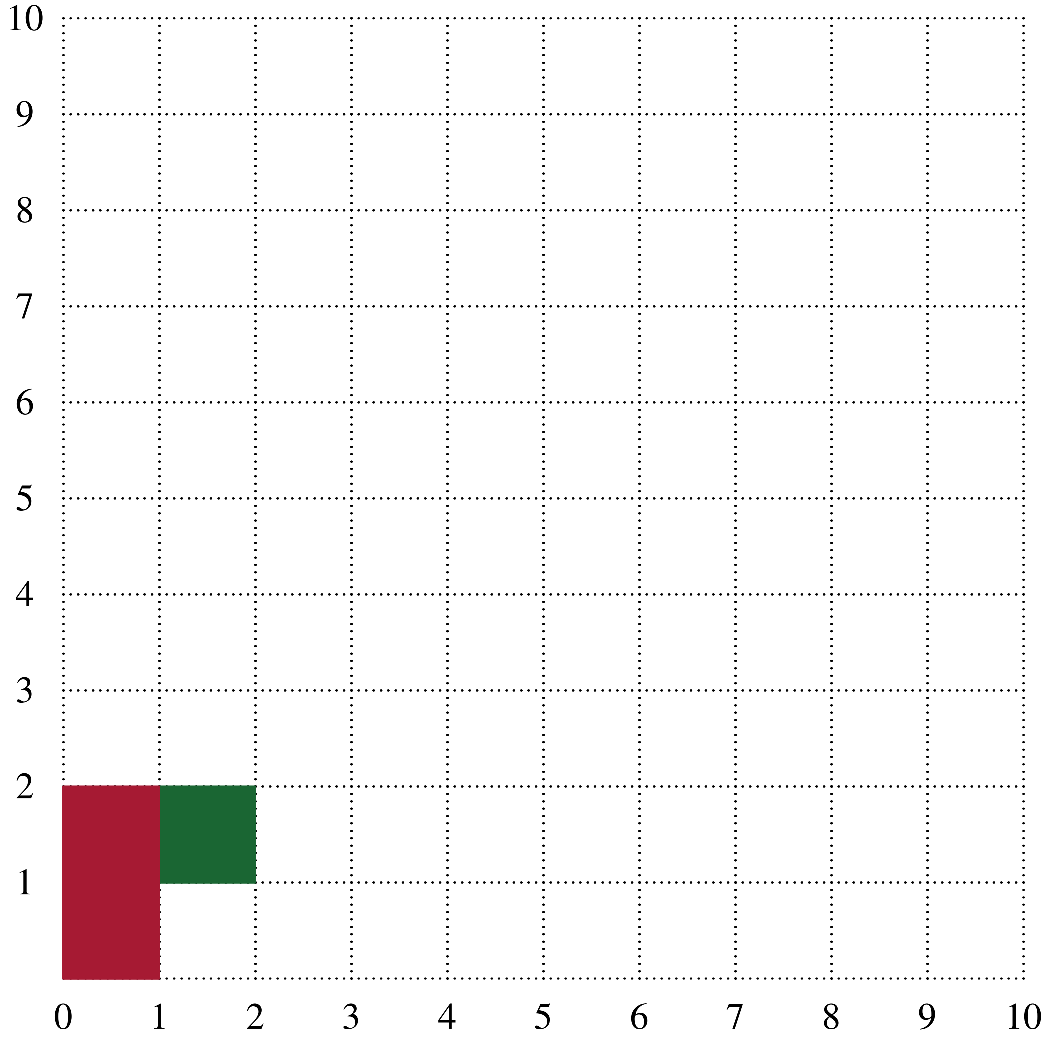
- area of box is:

-

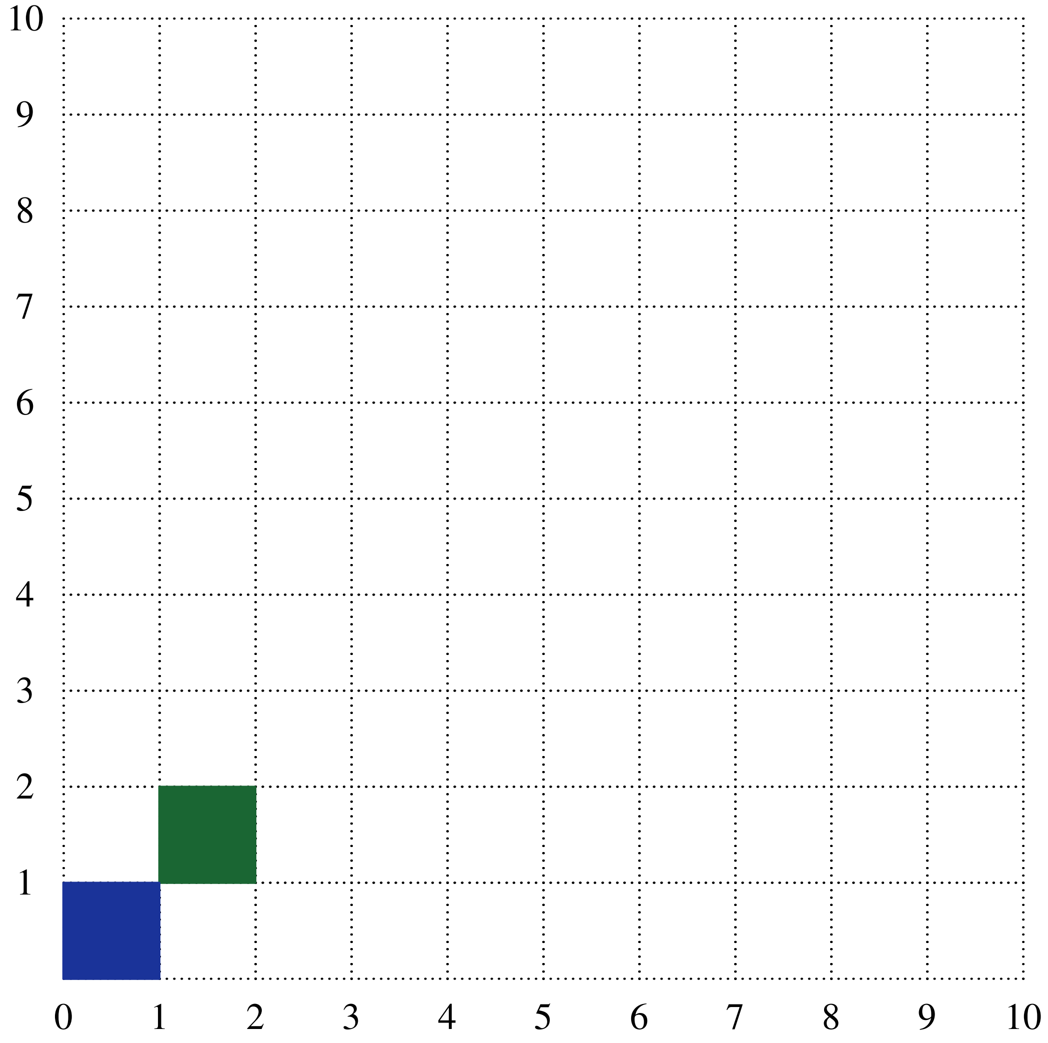
i	$x_i y_{i+1} - x_{i+1} y_i$	total
0	1×2 - 1×1	1
1	1×2 - 2×2	-2
2	2×1 - 2×2	-2
3	2×1 - 1×1	1
		-2

- area $= \frac{-2}{2} = -1$

Iteration 0.a of the area calculation



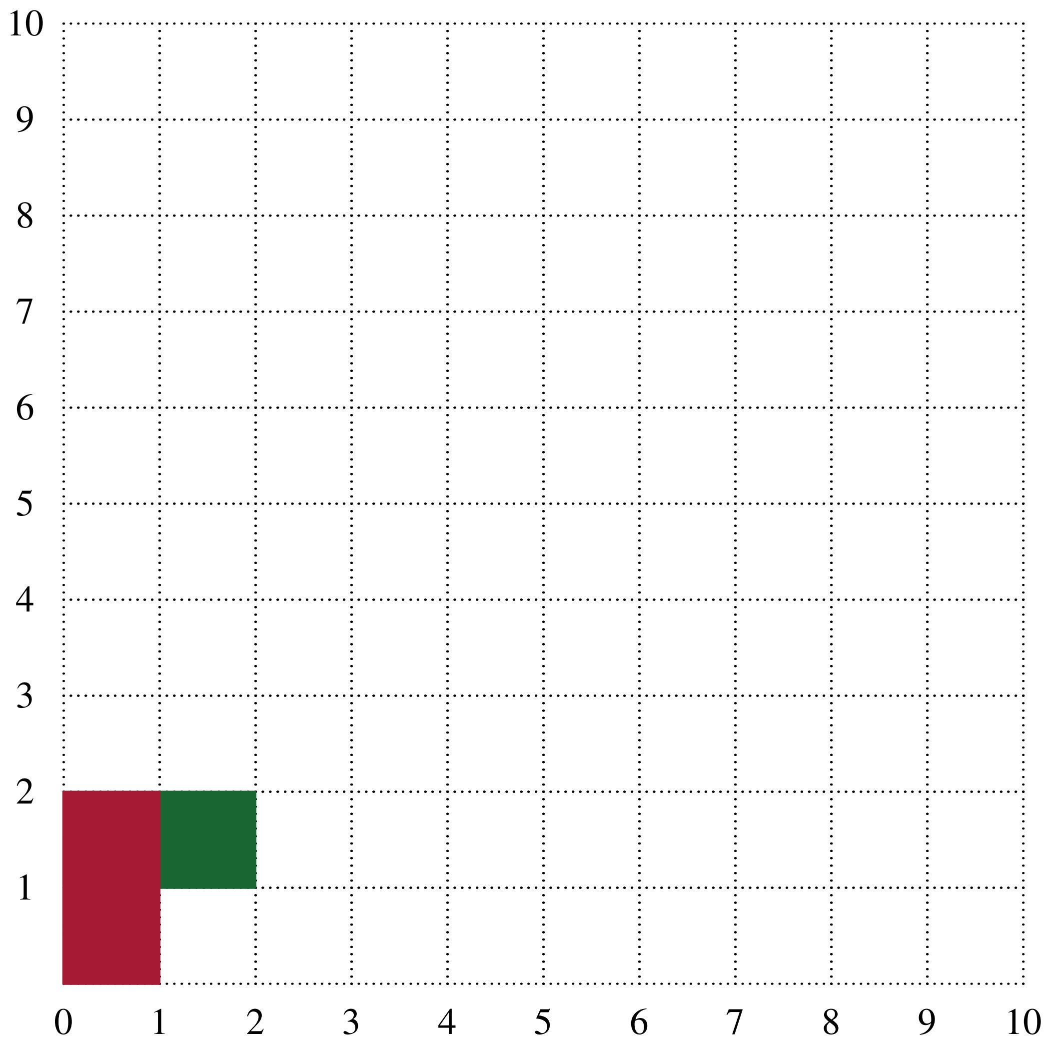
Iteration 0.b of the area calculation



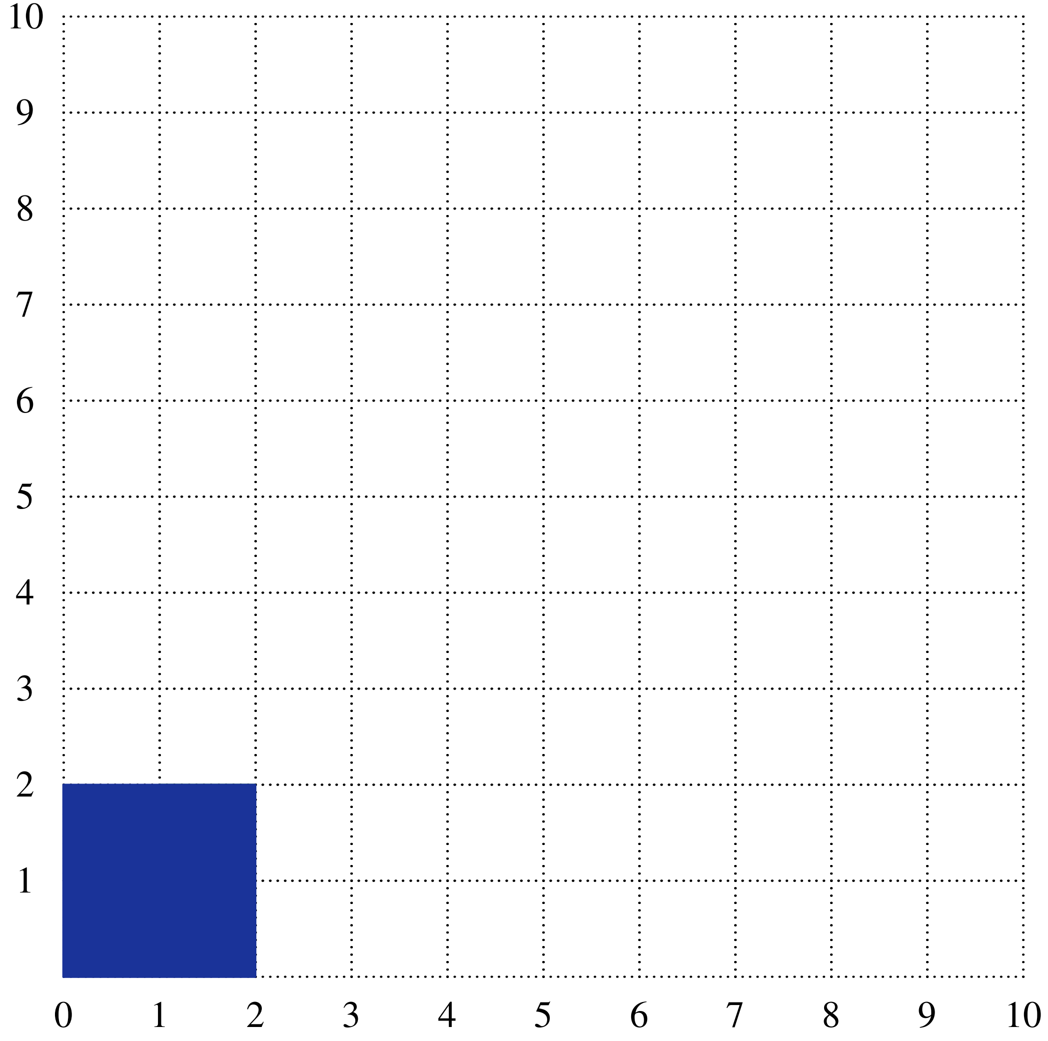
Iteration 0.b of the area calculation

- area of red box - blue box = 1

Iteration 1.a of the area calculation



Iteration 1.b of the area calculation



Iteration 1.b of the area calculation

- red box - blue box = -2
- exercise for the reader, complete the diagrams for the remaining iterations

Calculating the centre of gravity of a polygon

- C_x is calculated via:
- $$C_x = \frac{1}{6A} \sum_{i=0}^{n-1} (x_i + x_{i+1})(x_i y_{i+1} - x_{i+1} y_i)$$
- C_y is calculated via:
- $$C_y = \frac{1}{6A} \sum_{i=0}^{n-1} (y_i + y_{i+1})(x_i y_{i+1} - x_{i+1} y_i)$$

Calculating C of G in pge

- please see `c/twoDsim.c` and the functions `calculateCofG` and `calcArea`