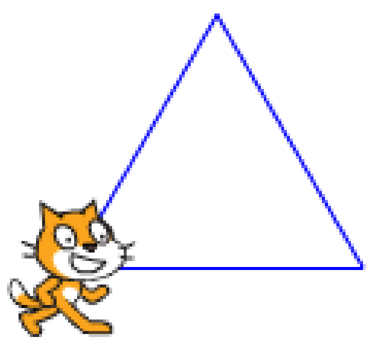
Ένας απλός και γρήγορος αλγόριθμος για την αποκοπή γραμμών στο Scratch

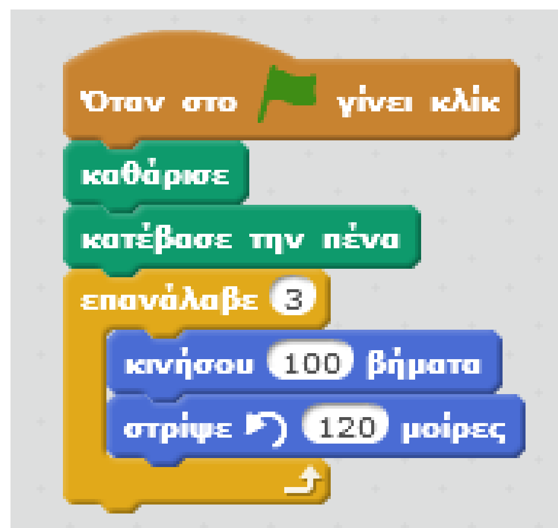
Ματθές Δημήτριος[1], Μαγουλάς Αντώνιος[2]

[1]Εκπαιδευτικός Πληροφορικής ΠΕ86, dimmat@gmail.com
[2]Εκπαιδευτικός Πληροφορικής ΠΕ03, amagul@yahoo.com

Σχεδίαση με πένα στο Scratch

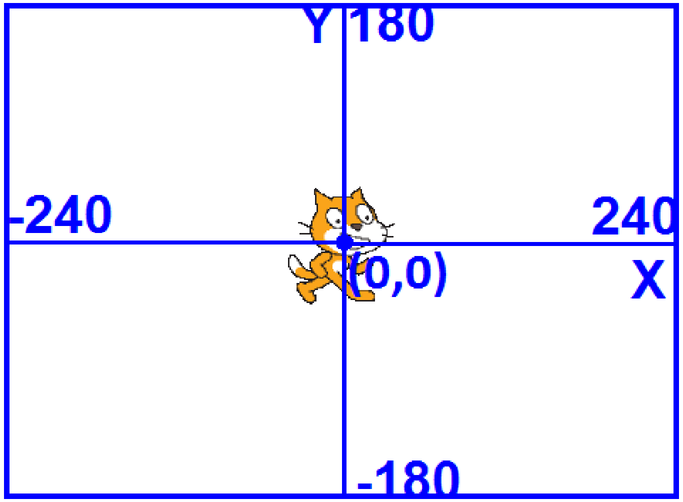
- Το Scratch διαθέτει μια ομάδα εντολών που ονομάζεται «**Σχεδιασμοί Πένας**».
- Κάθε μορφή μπορεί να χρησιμοποιήσει την πένα για να αφήσει ένα ίχνος όταν κινείται.
- Για παράδειγμα:





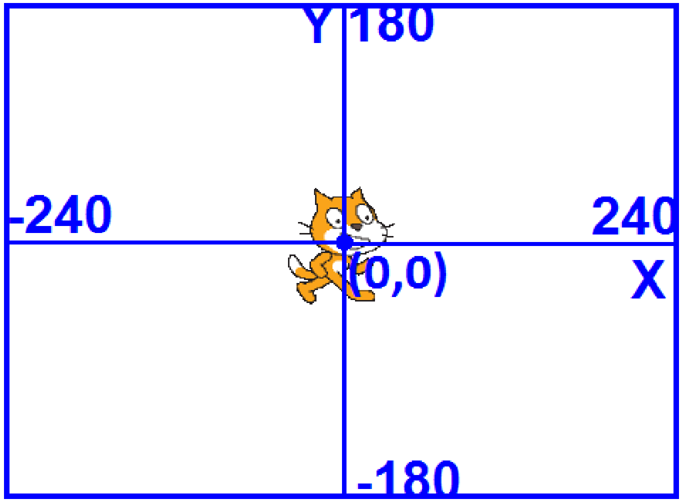
Οθόνη του Scratch και όρια

- Η οθόνη του Scratch είναι συγκεκριμένων διαστάσεων (480 εικονοστοιχεία πλάτος x 360 εικονοστοιχεία ύψος). Η αρχή των αξόνων θεωρείται το κέντρο (σημείο 0,0).
- Σε αυτή ο χρήστης μπορεί να κινεί τις διάφορες μορφές ή να σχεδιάζει σχήματα με τη βοήθεια της πένας.
- Δεν επιτρέπει στις μορφές ή στην πένα να «βγουν» εκτός των ορίων της οθόνης



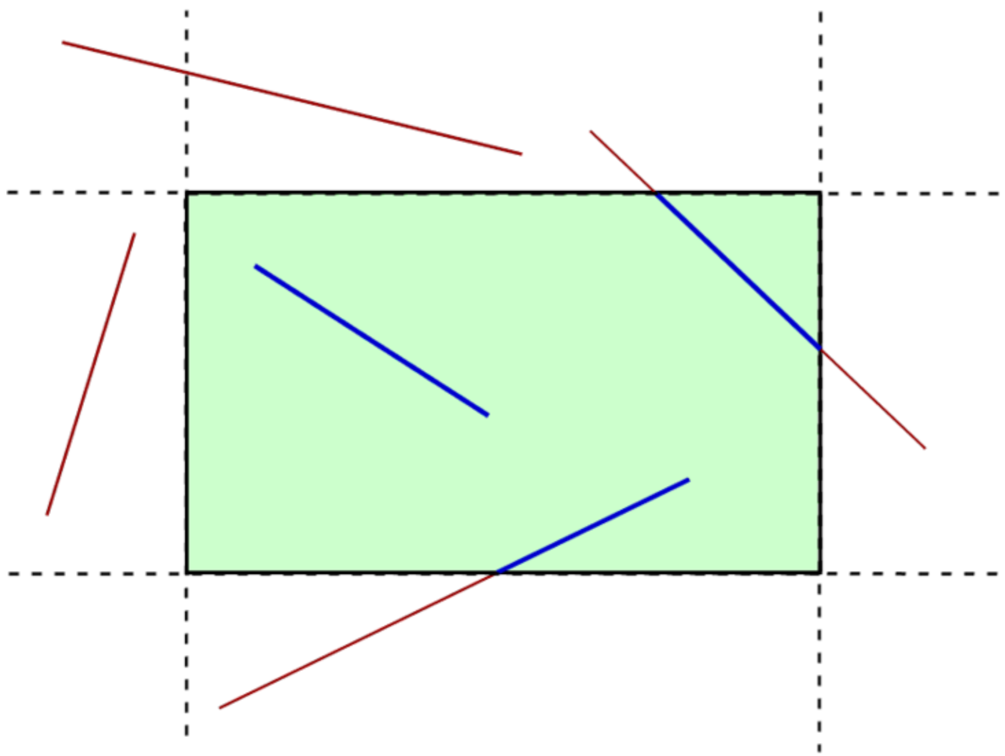
Μπορώ να ξεπεράσω τα όρια;

- Αν ο προγραμματιστής θελήσει να σχεδιάσει μία πάρα πολύ μεγάλη γραμμή η οποία εκτείνεται εκτός των ορίων, αυτό δεν είναι εφικτό με τις υπάρχουσες επιλογές.
- Τη λύση στο πρόβλημα της σχεδίασης μιας μεγάλης γραμμής ή γενικότερα ενός μεγάλου σχήματος έρχεται να δώσει η τεχνική που ονομάζεται «**αποκοπή γραμμής**» (line clipping).



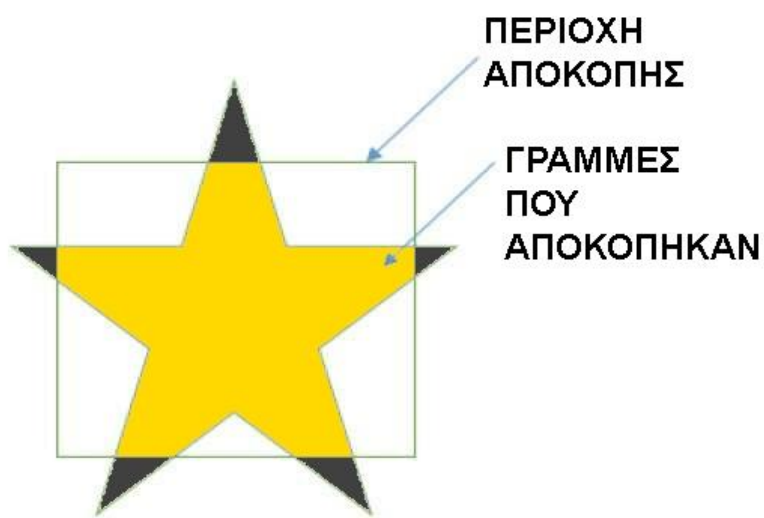
Τι είναι η αποκοπή γραμμής

- Ως αποκοπή γραμμής ορίζεται η διαδικασία αφαίρεσης των τμημάτων μιας γραμμής τα οποία βρίσκονται εκτός μιας επιθυμητής περιοχής (Hearn, 1997).



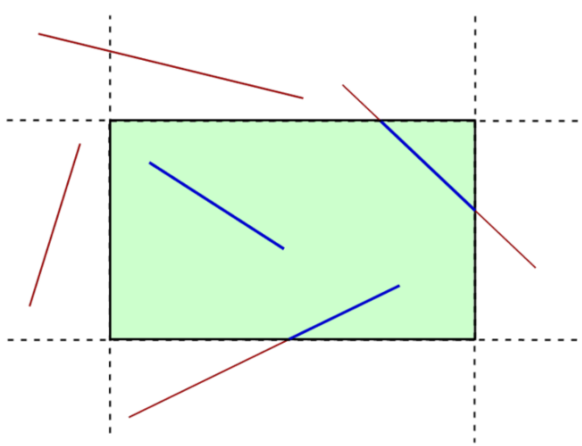
Τι είναι η αποκοπή γραμμής

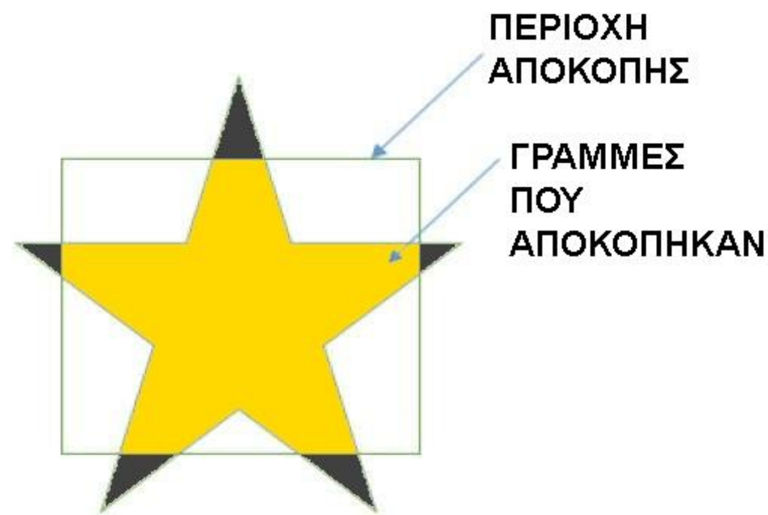
- Η διαδικασία της αφαίρεσης των περιττών τμημάτων γίνεται χρησιμοποιώντας τα μαθηματικά.
- Ο προγραμματιστής σχεδιάζει το τμήμα της γραμμής που βρίσκεται εντός των ορίων χρησιμοποιώντας:
  - είτε την εξίσωση της ευθείας y=a*x+b
  - είτε κάνοντας χρήση διανυσμάτων



Γνωστοί αλγόριθμοι αποκοπής γραμμών

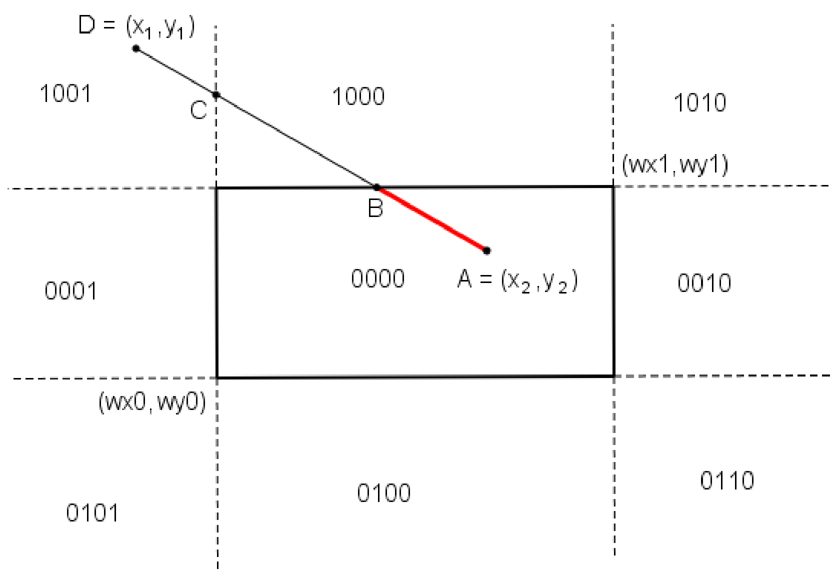
- Οι πιο γνωστοί αλγόριθμοι για την αποκοπή γραμμών είναι δύο:
  - Ο αλγόριθμος των **Cohen-Sutherland**
  - Ο αλγόριθμος των **Liang-Barsky**
- Πάνω σε αυτούς βασίστηκαν κι άλλοι αλγόριθμοι, όπως οι Cyrus-Beck, Nicholl-Lee-Nicholl, Fast clipping, Scala κλπ.





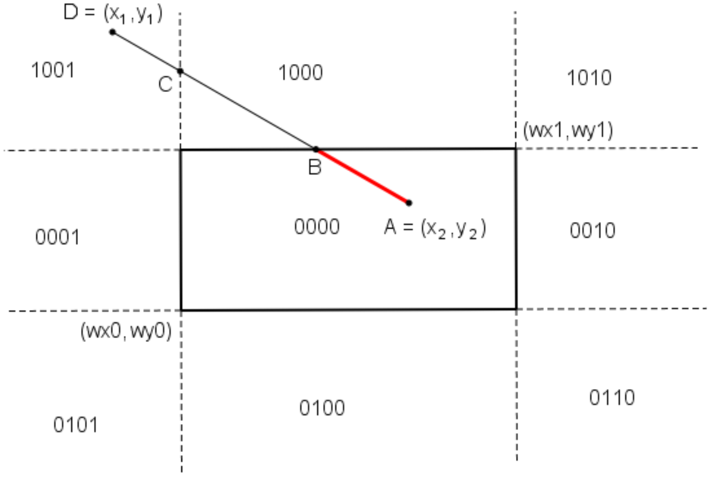
Αλγόριθμος Cohen-Sutherland

- Ο αλγόριθμος των Danny Cohen και Ivan Sutherland αναπτύχθηκε το 1967 κατά τη δημιουργία ενός εξομοιωτή πτήσεων.
- Θεωρείται ένας από τους πρώτους αλγόριθμους αποκοπής γραμμής στην ιστορία της σχεδίασης γραφικών.



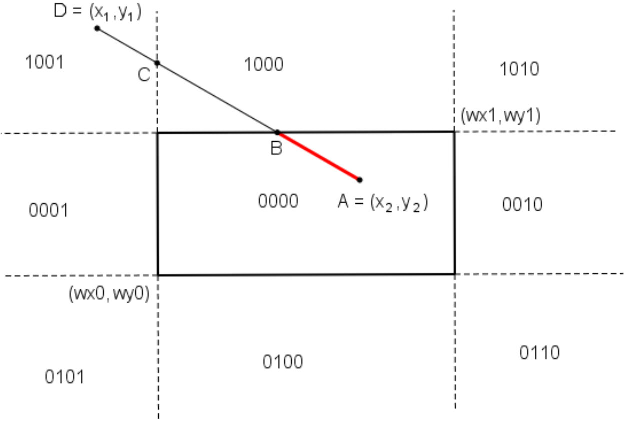
Αλγόριθμος Cohen-Sutherland

- Σύμφωνα με αυτόν, ο δισδιάστατος χώρος στον οποίο βρίσκεται η προς αποκοπή γραμμή χωρίζεται σε 9 τμήματα.
- Σε κάποια από αυτά εντοπίζονται τα δύο σημεία που ορίζουν την γραμμή.
- Ανάλογα με τις περιοχές αυτές, ο αλγόριθμος πραγματοποιεί πλήρη, μερική ή καθόλου σχεδίασή της (Foley, 1996).



Αλγόριθμος Cohen-Sutherland

- Για να εντοπιστούν οι περιοχές όπου βρίσκονται η αρχή και το τέλος της γραμμής και να σχεδιαστεί η αποκομμένη γραμμή απαιτούνται πράξεις «λογικού ΚΑΙ» (bitwise AND).
- Η πράξη του «λογικού ΚΑΙ» δεν υπάρχει στο Scratch. Ο προγραμματιστής θα πρέπει να την κατασκευάσει (δημιουργία υπορουτίνας). Η κατασκευή της όμως επιβαρύνει κατά πολύ σε ταχύτητα τον αλγόριθμο.



Αλγόριθμος Liang-Barsky

- Ο αλγόριθμος των You-Dong Liang και Brian Barsky χρησιμοποιεί την εξίσωση της ευθείας καθώς και κάποιες ανισότητες για να βρει την περιοχής αποκοπής και να προσδιορίσει τα σημεία τομής της με την προς σχεδίαση γραμμή (Liang & Barsky, 1984).

- Parametric Equation of the line

Line (x_1, y_1) *to* (x_2, y_2)

Consider t: range from 0 to 1

At Start of the line (x_1, y_1) t=0

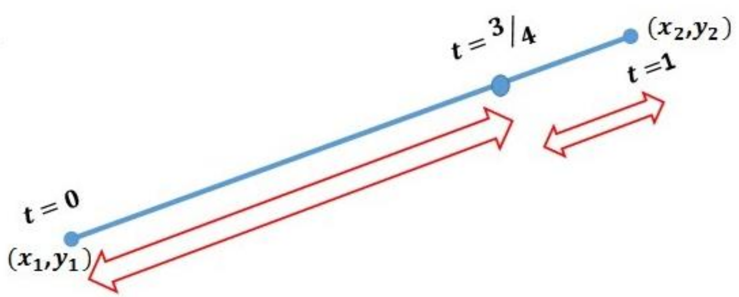
At End of the line (x_2, y_2) t=1

At ${}^3/_4{}^{th}$ of the line path t=${}^3/_4$

The location is

$x = {}^1/_4\, x_1 + {}^3/_4\, x_2$

$y = {}^1/_4\, y_1 + {}^3/_4\, y_2$



Αλγόριθμος Liang-Barsky

- Όμως, κι αυτός ο αλγόριθμος στην υλοποίησή του στο Scratch απαιτεί έναν σχετικά μεγάλο αριθμό συγκρίσεων προκειμένου να σχεδιαστεί η γραμμή με αποτέλεσμα να μην είναι ιδιαίτερα αποδοτικός.

- Parametric Equation of the line

Line (x_1, y_1) to (x_2, y_2)

Consider t: range from 0 to 1

At Start of the line (x_1, y_1) t=0

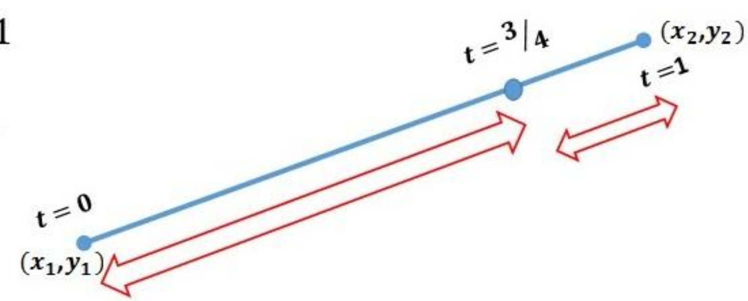
At End of the line (x_2, y_2) t=1

At $^3/_4{}^{th}$ of the line path t=$^3/_4$

The location is

x=$^1/_4\, x_1 + {}^3/_4\, x_2$

y=$^1/_4\, y_1 + {}^3/_4\, y_2$

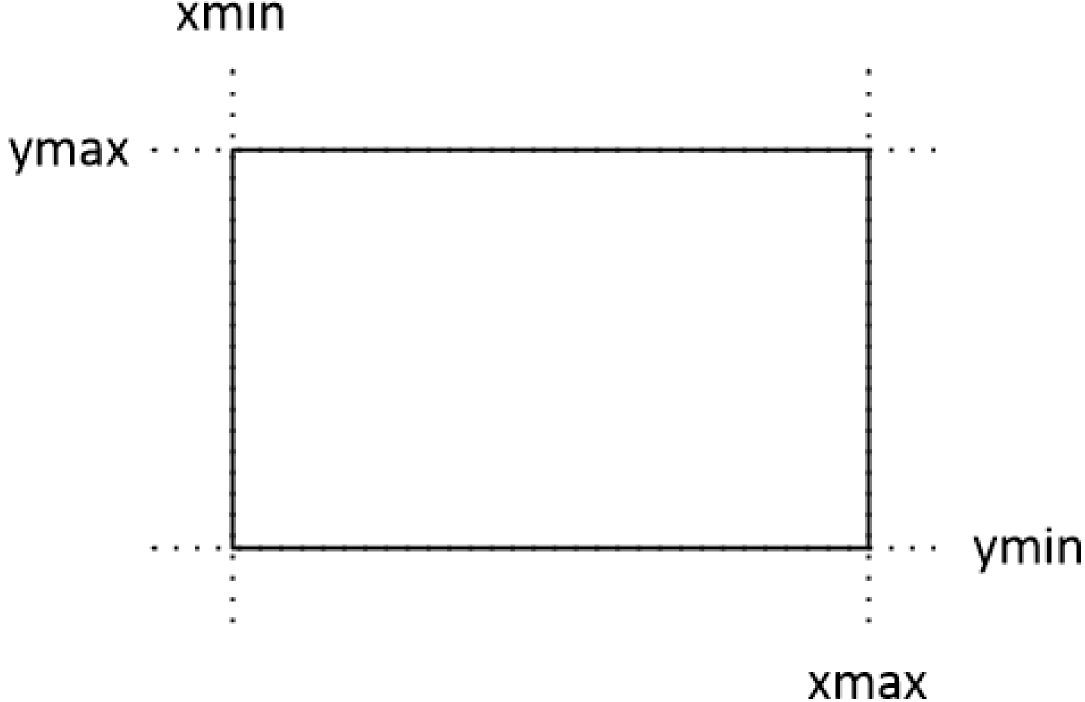


Προτεινόμενος αλγόριθμος

- Τις δυσκολίες των δύο γνωστών αλγορίθμων αποκοπής γραμμής στο Scratch φαίνεται να ξεπερνά ο επόμενος προτεινόμενος αλγόριθμος.
- Στοχεύει στην απλότητα και την ταχύτητα και πραγματοποιεί τις απαραίτητες συγκρίσεις προκειμένου να προσδιορίσει αν η αρχή και το τέλος της γραμμής βρίσκονται εντός της περιοχής αποκοπής.
- Επιπλέον, χρησιμοποιεί ελάχιστες μεταβλητές για λόγους απλότητας και ευκολίας.

Θεωρητικό υπόβαθρο

- Θεωρούμε ότι θέλουμε να αποκόψουμε μια γραμμή εντός μιας ορθογώνιας περιοχής η οποία προσδιορίζεται από τα εξής σημεία: (xmin, ymax) και (xmax, ymin), όπως στην εικόνα:



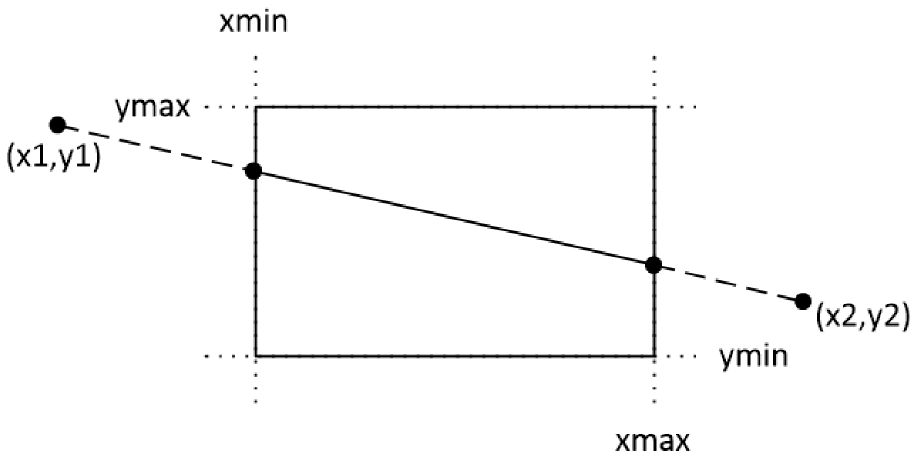
Θεωρητικό υπόβαθρο

- Έστω (x1, y1) και (x2,y2) δύο γνωστά σημεία που προσδιορίζουν τη γραμμή που θέλουμε να σχεδιάσουμε. Σύμφωνα με τα μαθηματικά, η κλίση m της γραμμής είναι πάντα σταθερή και προσδιορίζεται από τον λόγο:

$$\mathrm{m} = \frac{(\mathrm{y2} - \mathrm{y1})}{(\mathrm{x2} - \mathrm{x1})} \Rightarrow$$

$$\Rightarrow \mathrm{y2} - \mathrm{y1} = \mathrm{m} * (\mathrm{x2} - \mathrm{x1}) \Rightarrow$$

$$\Rightarrow \mathrm{y2} = \mathrm{m} * (\mathrm{x2} - \mathrm{x1}) + \mathrm{y1}$$

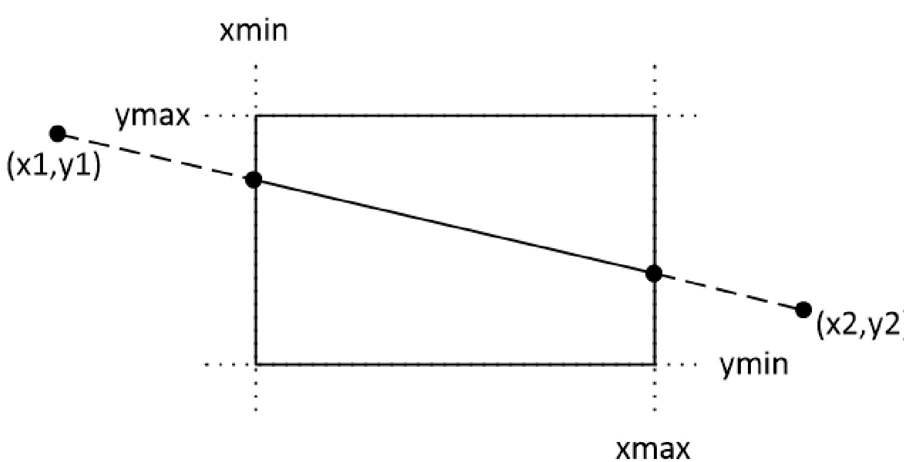


Θεωρητικό υπόβαθρο

- Για ένα οποιοδήποτε σημείο (x,y) της γραμμής, ο παραπάνω τύπος μπορεί να γραφεί υπό τη μορφή εξίσωσης ως εξής:

$$y = m * (x - x1) + y1 \Rightarrow$$

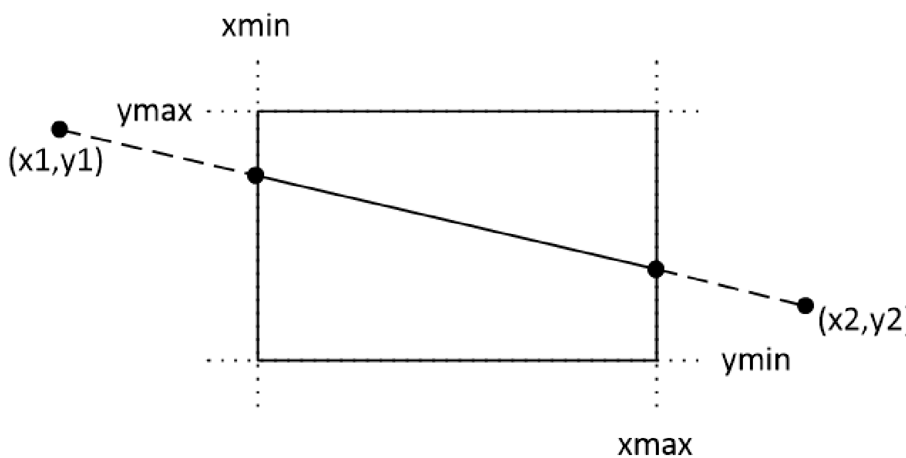
$$\boxed{\Rightarrow y = \frac{(y2 - y1)}{(x2 - x1)} \cdot (x - x1) + y1}$$



Θεωρητικό υπόβαθρο

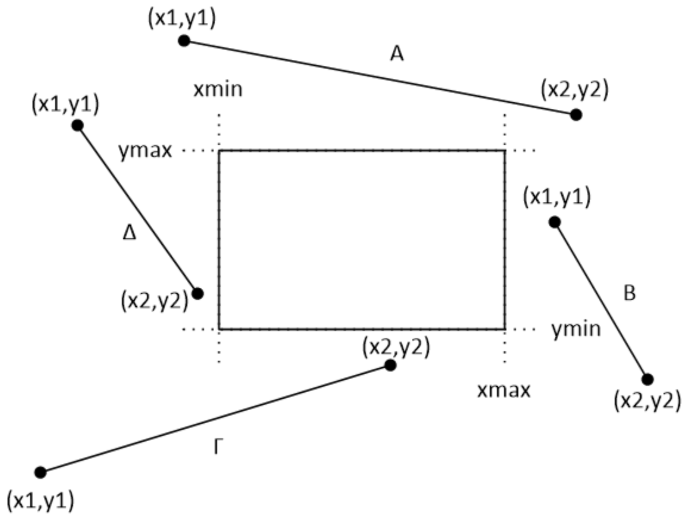
- Ομοίως, αν λύσουμε ως προς x, η προηγούμενη εξίσωση γίνεται:

$$\boxed{x = \frac{(x2 - x1)}{(y2 - y1)} \cdot (y - y1) + x1}$$



1ο βήμα του προτεινόμενου αλγορίθμου

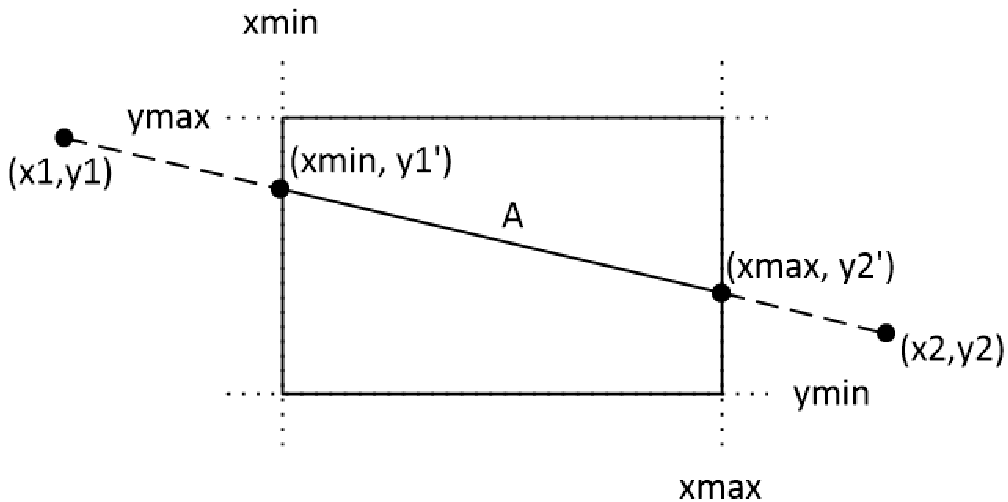
- Έστω ότι η προς αποκοπή γραμμή προσδιορίζεται από τα σημεία (x1,y1) και (x2,y2).
- Κατά το πρώτο βήμα του αλγορίθμου ελέγχεται αν και τα δύο σημεία της γραμμής βρίσκονται εκτός της περιοχής αποκοπής και ταυτόχρονα στο ίδιο τμήμα (δηλαδή πάνω, κάτω, δεξιά, αριστερά).
- Αν κάτι από τα παρακάτω συμβαίνει τότε ολόκληρη η γραμμή είναι εκτός της περιοχής αποκοπής και ο αλγόριθμος δεν την σχεδιάζει.



x1<xmin ΚΑΙ x2<xmin	(η γραμμή βρίσκεται αριστερά της περιοχής αποκοπής)
x1>xmax ΚΑΙ x2>xmax	(η γραμμή βρίσκεται δεξιά της περιοχής αποκοπής)
y1<ymin ΚΑΙ y2<ymin	(η γραμμή βρίσκεται κάτω από την περιοχή αποκοπής)
y1>ymax ΚΑΙ y2>ymax	(η γραμμή βρίσκεται πάνω από την περιοχή αποκοπής)

2ο βήμα του προτεινόμενου αλγορίθμου

- Στο δεύτερο βήμα, ο αλγόριθμος συγκρίνει τις συντεταγμένες των δύο σημείων με τα όρια της περιοχής αποκοπής.
- Συγκρίνει κάθε ένα από τα x1 και x2 με τα όρια xmin και xmax καθώς και κάθε ένα από τα y1 και y2 με τα όρια ymin και ymax.
- Αν κάποια από τις παραπάνω συντεταγμένες είναι εκτός ορίων τότε το συγκεκριμένο όριο (xmin, xmax, ymin, ymax) χρησιμοποιείται στην εξίσωση που προσδιορίζει την γραμμή προκειμένου αλλάξουν τα σημεία που την προσδιορίζουν για να επιτευχθεί αποκοπή.

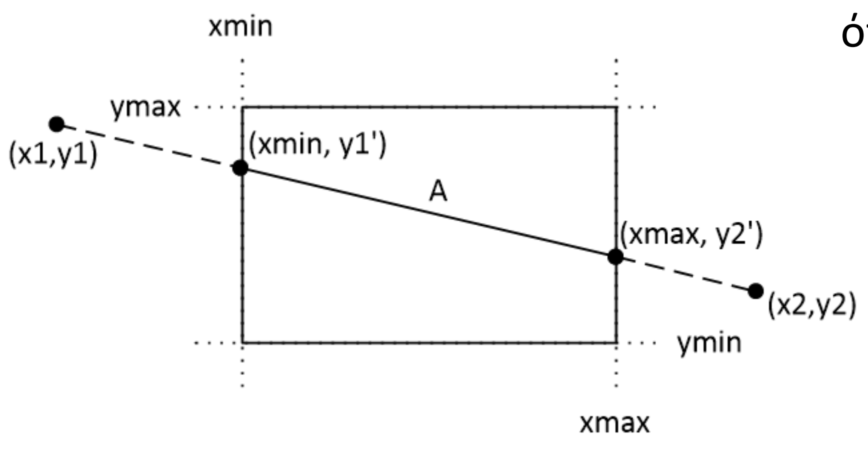


2ο βήμα του προτεινόμενου αλγορίθμου

Πιο συγκεκριμένα:

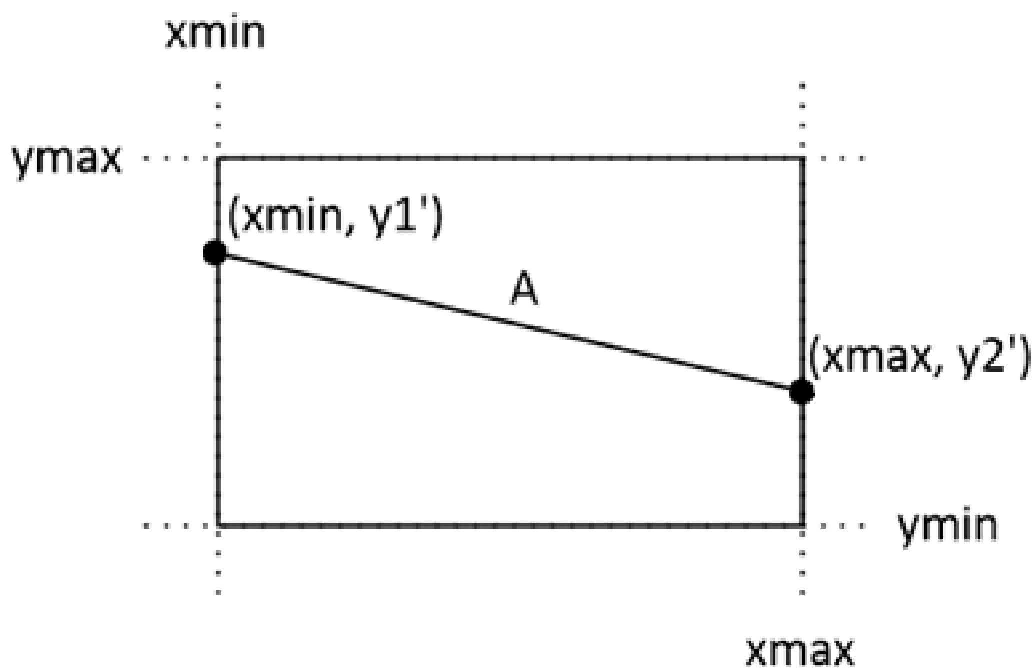
Αν το x_i < xmin τότε	x_i=xmin	$y_i = \frac{(y2-y1)}{(x2-x1)} \cdot (x\min - x1) + y1$
Αν το x_i > xmax τότε	xi=xmax	$y_i = \frac{(y2-y1)}{(x2-x1)} \cdot (x\max - x1) + y1$
Αν το y_i < ymin τότε	yi=ymin	$x_i = \frac{(x2-x1)}{(y2-y1)} \cdot (y\min - y1) + x1$
Αν το y_i > ymax τότε	yi=ymax	$x_i = \frac{(x2-x1)}{(y2-y1)} \cdot (y\max - y1) + x1$

όπου i: από 1 έως 2.



3ο βήμα του προτεινόμενου αλγορίθμου

- Το τρίτο και τελευταίο βήμα του αλγορίθμου συγκρίνει τα νέα σημεία που προέκυψαν μετά τις αλλαγές.
- Αν αυτά είναι εντός των ορίων της περιοχής αποκοπής τότε σχεδιάζει τη γραμμή που προσδιορίζεται από αυτά.



Ψευδοκώδικας προτεινόμενου αλγόριθμου

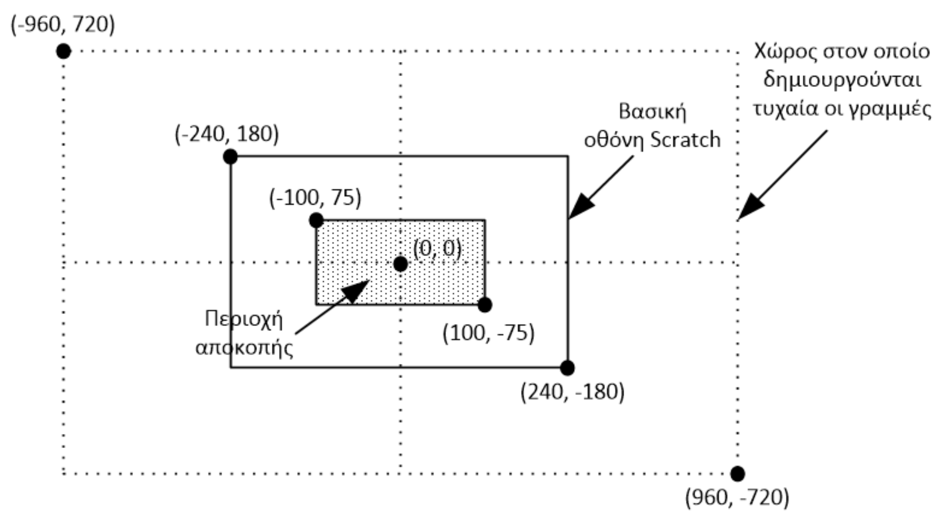
```
Αλγόριθμος Scratch_Line_Clipping
Δεδομένα // x[2], y[2], xmin, ymax, xmax, ymin //
Αν όχι(x[1]<xmin και x[2]<xmin) και όχι(x[1]>xmax και x[2]>xmax) τότε
 Αν όχι(y[1]<ymin και y[2]<ymin) και όχι(y[1]>ymax και y[2]>ymax) τότε
  i<-1
  Αρχή_επανάληψης
   Αν x[i] < xmin τότε
     x[i]<-xmin
     y[i]<- ((y[2]-y[1])/(x[2]-x[1])*(xmin-x[1])+y[1]
   Τέλος_αν
   Αν x[i] > xmax τότε
     x[i]<-xmax
     y[i]<- ((y[2]-y[1])/(x[2]-x[1])*(xmax-x[1])+y[1]
   Τέλος_αν
 Αν y[i] < ymin τότε
     y[i]<-ymin
     x[i]<- ((x[2]-x[1])/(y[2]-y[1])*(ymin-y[1])+x[1]
   Τέλος_αν
      Αν y[i] > ymax τότε
     y[i]<-ymax
     x[i]<- ((x[2]-x[1])/(y[2]-y[1])*(ymax-y[1])+x[1]
   Τέλος_αν
   i<-i+1
  Μέχρις_ότου i>2

  Αν όχι(x[1]<xmin και x[2]<xmin) και όχι(x[1]>xmax και x[2]>xmax) τότε
   Αν όχι(y[1]<ymin και y[2]<ymin) και όχι(y[1]>ymax και y[2]>ymax) τότε
    Σχεδίασε_γραμμή(x[1],y[1],x[2],y[2])
   Τέλος_αν
  Τέλος_αν

 Τέλος_αν
Τέλος_αν
Τέλος Scratch_Line_Clipping
```

Συγκρίνοντας τους αλγορίθμους

- Για να προσδιορίσουμε την αποδοτικότητα του προτεινόμενου αλγορίθμου αποφασίσαμε να τον συγκρίναμε με τους δύο πιο γνωστούς αλγόριθμους αποκοπής γραμμών: τον αλγόριθμο των Cohen-Sutherland και τον αλγόριθμο των Liang-Barsky.
- Όλοι οι αλγόριθμοι υλοποιήθηκαν έχοντας ως βασικούς άξονες την απλότητα και την αποδοτικότητα και καταβλήθηκε προσπάθεια προκειμένου να έχουν την ίδια περίπου δομή.

Προϋποθέσεις πειράματος

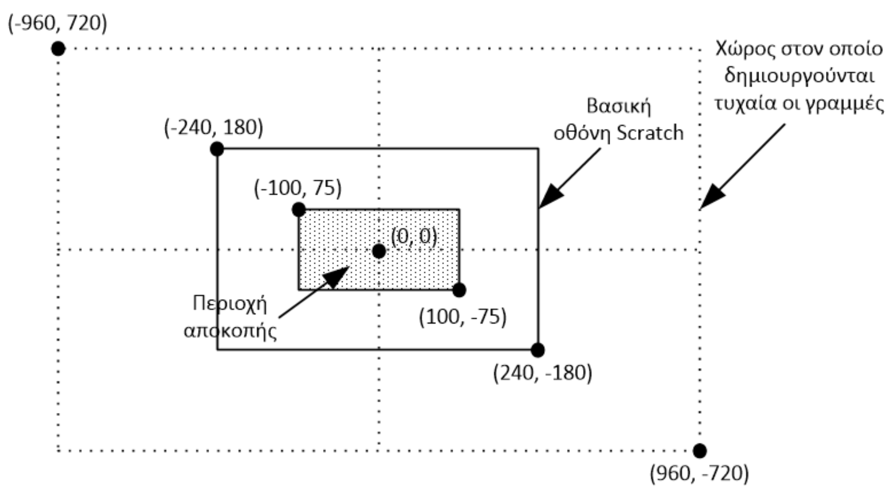
- Το πείραμα για τη σύγκριση ήταν το εξής: Κάθε ένας από τους αλγορίθμους θα έπρεπε να δημιουργεί 10.000 τυχαίες γραμμές σε ένα δισδιάστατο χώρο 4πλάσιο από τη βασική οθόνη σχεδίασης του Scratch.
- Ένας τέτοιος χώρος προσδιορίζεται από τα σημεία (-960, 720) και (960,-720). Η περιοχή αποκοπής θα έπρεπε να βρίσκεται στο κέντρο της οθόνης σχεδίασης του Scratch και ορίζεται από τα σημεία (-100, 75) και (100,-75)
- Η γραμμές θα έπρεπε να εκτείνονται τυχαία, οπουδήποτε μέσα στον μεγάλο χώρο, να αποκόπτονται τα περιττά τμήματά τους και να σχεδιάζονται μόνο τμήματα εκείνα που βρίσκονται εντός της περιοχής αποκοπής



Προϋποθέσεις πειράματος

- Ο χρόνος που απαιτείται κάθε φορά για τη σχεδίαση των 10.000 γραμμών καταγράφεται για κάθε έναν από τους αλγορίθμους.
- Η διαδικασία επαναλαμβάνεται 10 φορές και στο τέλος υπολογίζεται ο μέσος χρόνος εκτέλεσης.

Αλγόριθμος	Διεύθυνση URL
Cohen-Sutherland	**https://scratch.mit.edu/projects/166917422**
Liang-Barsky	**https://scratch.mit.edu/projects/166820820**
Προτεινόμενος	**https://scratch.mit.edu/projects/166877443**



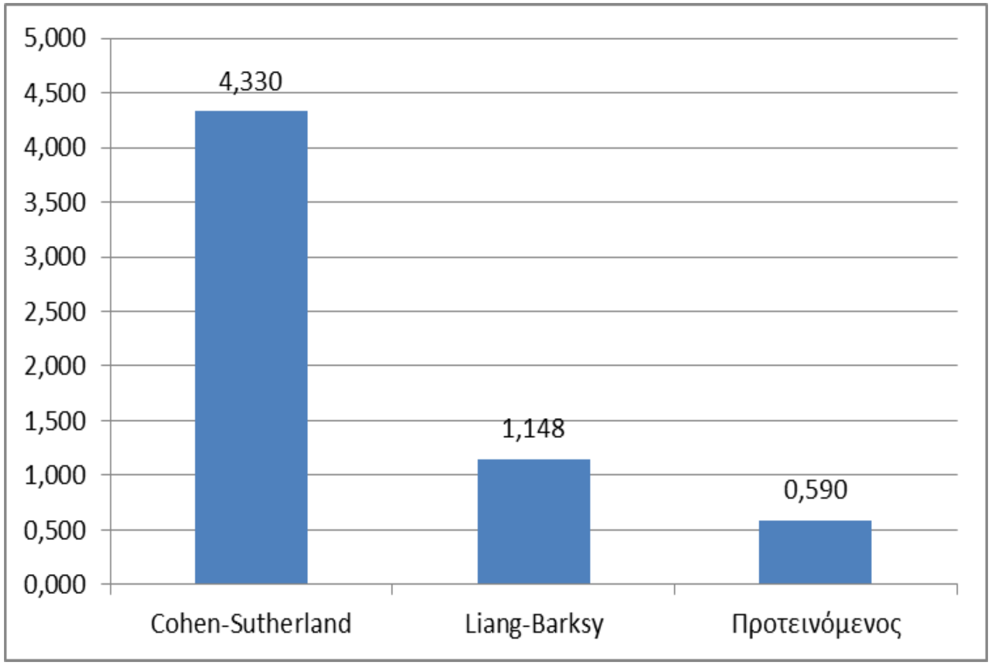
Αποτελέσματα

- Στον πίνακα φαίνονται οι χρόνοι εκτέλεσης των συγκρινόμενων αλγορίθμων:

Εκτέλεση	Cohen-Sutherland (sec)	Liang-Barsky (sec)	Προτεινόμενος Αλγόριθμος (sec)
1	4,245	1,159	**0,625**
2	4,279	1,149	**0,632**
3	4,456	1,159	**0,592**
4	4,364	1,135	**0,618**
5	4,275	1,132	**0,592**
6	4,352	1,142	**0,427**
7	4,347	1,167	**0,630**
8	4,356	1,132	**0,620**
9	4,295	1,143	**0,577**
10	4,326	1,159	**0,587**
Μέσος χρόνος :	**4,330**	**1,148**	**0,590**

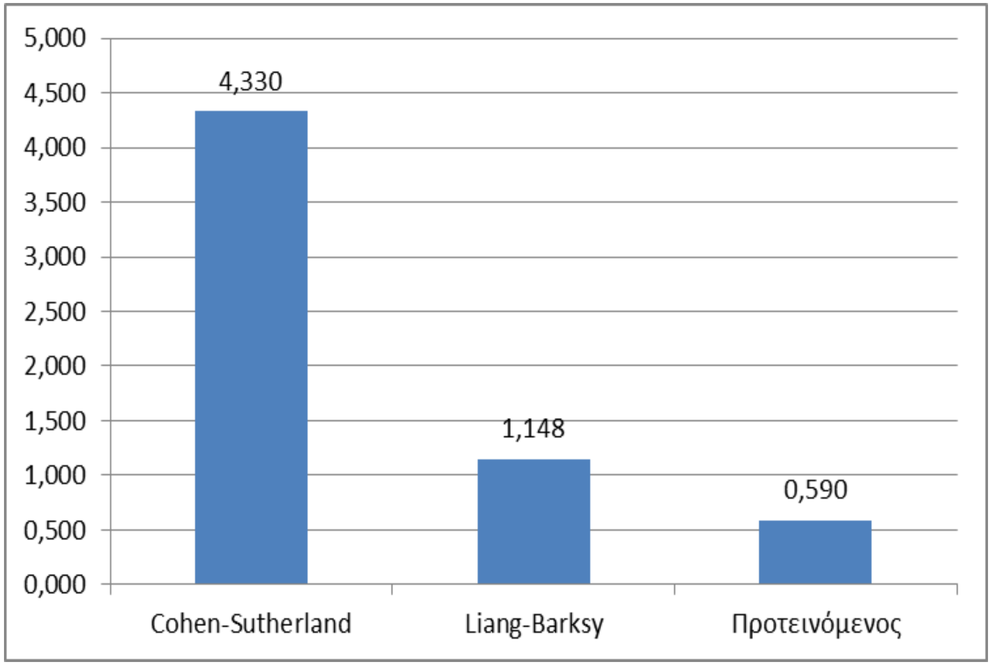
Συμπεράσματα

- Συγκρίνοντας τους χρόνους μεταξύ τους παρατηρούμε ότι στο Scratch:
  - Ο προτεινόμενος αλγόριθμος είναι περίπου 10 φορές πιο γρήγορος από τον αλγόριθμο των Cohen-Sutherland
  - Σχεδόν 2 φορές πιο γρήγορος από τον αλγόριθμο των Liang-Barsky



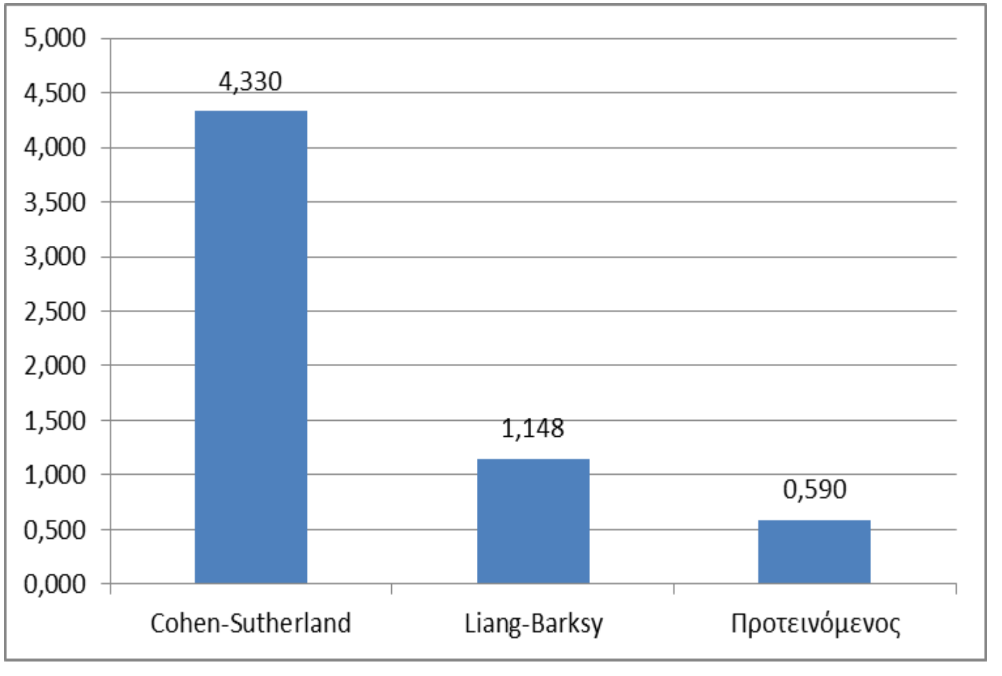
Συμπεράσματα

- Ο αλγόριθμος των Cohen-Sutherland είναι πιο αργός στην υλοποίησή του στο Scratch διότι οι πράξεις του λογικού ΚΑΙ (bitwise AND) που απαιτούνται σε κάθε επανάληψη καθυστερούν σημαντικά την εκτέλεση του.



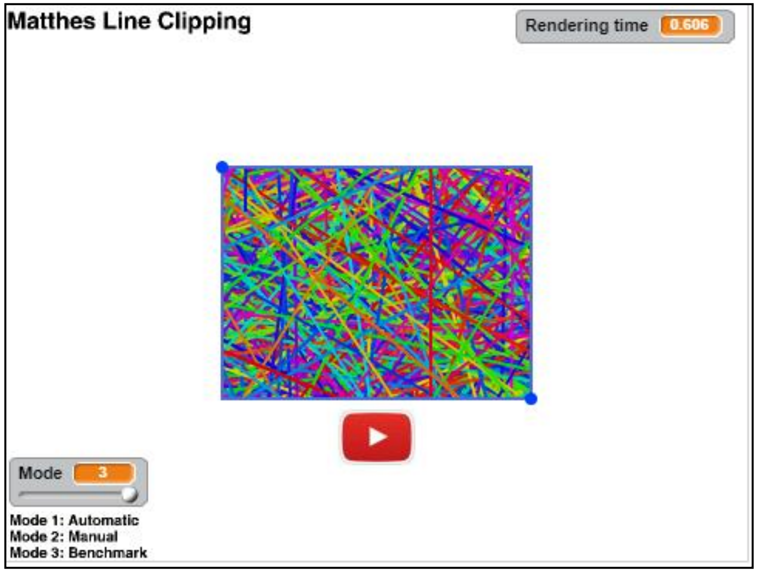
Συμπεράσματα

- Ο αλγόριθμος των Liang-Barsky δείχνει αρκετά πιο γρήγορος σε σχέση με τον Cohen-Sutherland και μοιάζει να πλησιάζει τον προτεινόμενο αλγόριθμο σε ταχύτητα.
- Είναι όμως πιο πολύπλοκος στην υλοποίησή του και ελαφρώς πιο δυσνόητος διότι εμπεριέχει δυσκολότερες μαθηματικές έννοιες.



Συμπεράσματα

- Αν το παραπάνω πείραμα επαναληφθεί με διαφορετικές παραμέτρους (π.χ. σχεδίαση περισσότερων γραμμών, διαφορετικός δισδιάστατος χώρος δημιουργίας των γραμμών, διαφορετική περιοχή αποκοπής, άλλο υπολογιστικό σύστημα) τα αποτελέσματα είναι ανάλογα.



Αυτοαξιολόγηση

- Ο προτεινόμενος αλγόριθμος αποκοπής γραμμών, στην υλοποίησή του στο Scratch, δύναται να αποκόψει και να σχεδιάσει 10.000 γραμμές σε χρόνο κάτω του 1 δευτερολέπτου
- Είναι αρκετά απλός.
- Αίρει τους περιορισμούς οθόνης του Scratch.
- Δίνει τη δυνατότητα σε εκπαιδευτικούς να διδάξουν την έννοια της αποκοπής γραμμής πολύ απλά και με πολύ γρήγορα αποτελέσματα.

Ευχαριστούμε για την προσοχή σας!

Ερωτήσεις;