**Γραφικά και εικονική πραγματικότητα**

**Εργαστήριο 1**

Ονοματεπώνυμο: Χρήστος Σούρμπης

ΑΜ: 228518

Task 1)

Η απάντηση αυτού του ερωτήματος δίδεται μέσω του παρακάτω κώδικα:

glGenBuffers(1, &colorsVBO); // δημιουργία του καινούριου buffer

glBindBuffer(GL\_ARRAY\_BUFFER, colorsVBO); //σύνδεση του buffer με το αντικείμενο colorsVBO

glBufferData(GL\_ARRAY\_BUFFER, 9 \* sizeof(GLfloat), colors, GL\_STATIC\_DRAW);

// δήλωση μεγέθους του buffer, έχουμε βάλει 9 διότι σχεδιάζουμε τρίγωνα και κάθε τρίγωνο έχει 3 σημεία και κάθε σημείο περιγράφεται από τρεις αριθμούς

glVertexAttribPointer(1, 3, GL\_FLOAT, GL\_FALSE, 0, NULL); //δήλωση του vertex σαν attribute του αντικειμένου triangleVAO

και αφού συμπληρώσουμε τις εντολές στους shaders όπως δίνονται στην αναφορά.

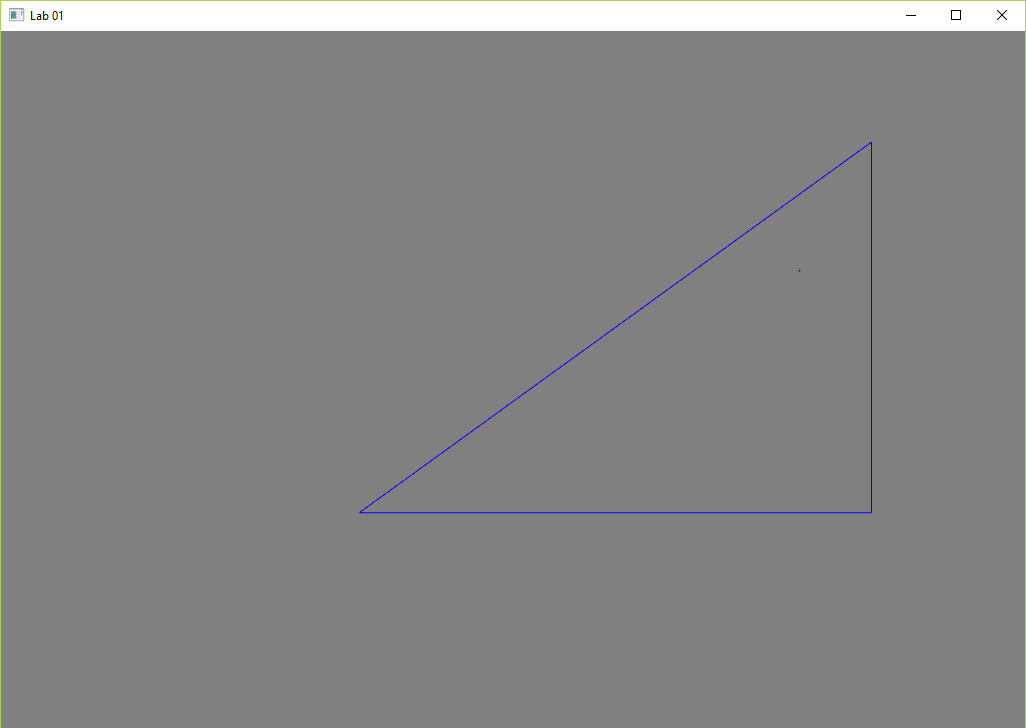
Σε αυτό το σημείο το vertex buffer object και το μόνο που μένει είναι να τον ενεργοποιήσουμε όταν θέλουμε να τον χρησιμοποιήσουμε:

glEnableVertexAttribArray(1);

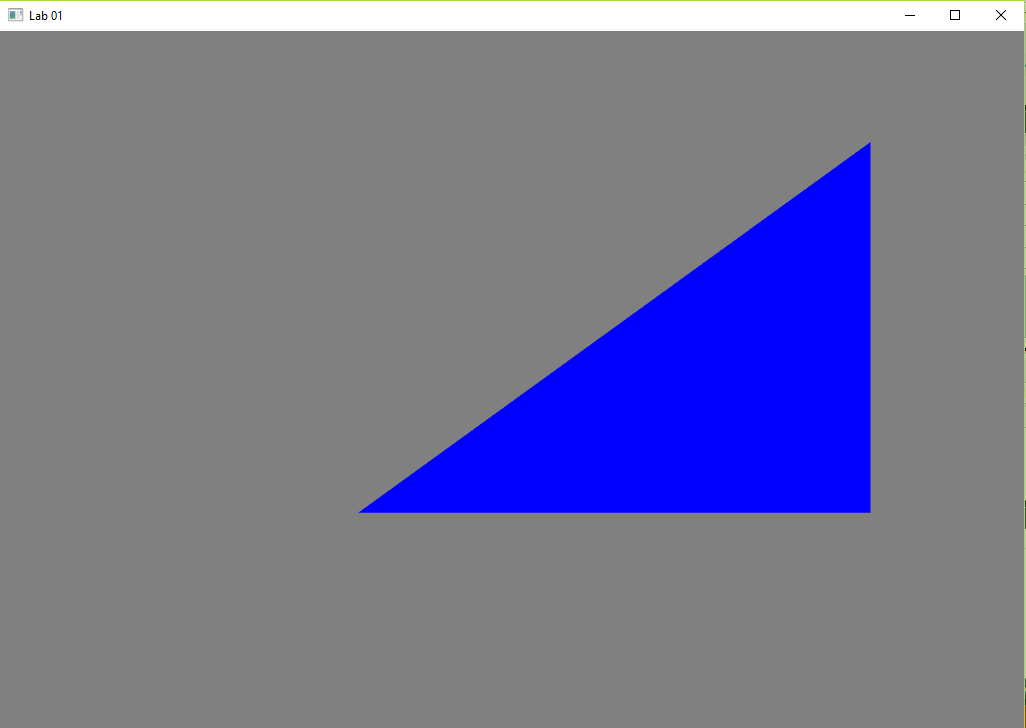
Καθώς και να το αποδεσμεύσουμε όταν τελειώσουμε με αυτόν:

glDeleteBuffers(1, &colorsVBO);

Task 2)

Με αυτήν την εντολή λέμε στο πρόγραμμα μας να ζωγραφίσει το τρίγωνο είτε μόνο το περίγραμμά του: glPolygonMode(GL\_FRONT\_AND\_BACK, GL\_LINE); 

είτε το περίγραμμα και το εσωτερικό του glPolygonMode(GL\_FRONT\_AND\_BACK, GL\_FILL);

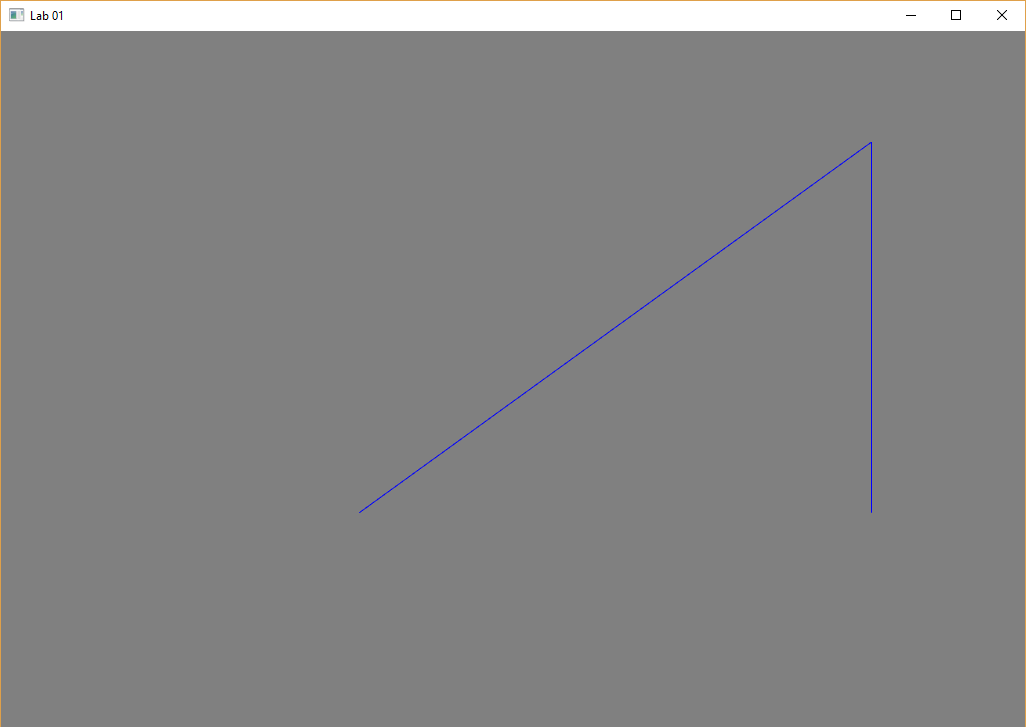


Task 3)

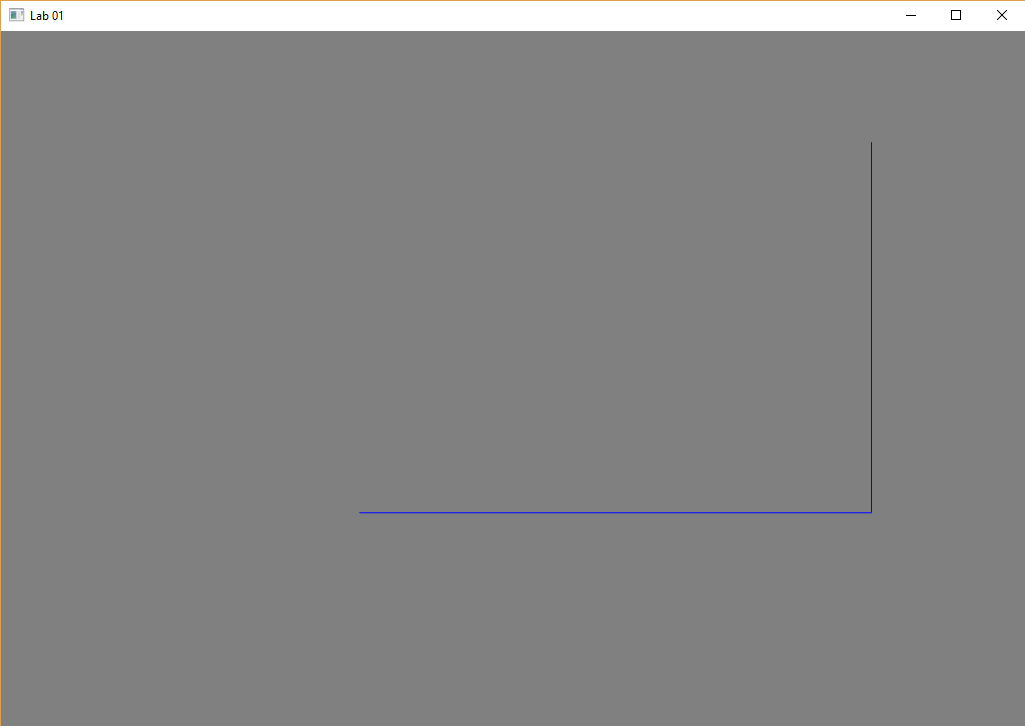
Σε αυτό το ερώτημα δοκιμάζουμε τα modes GL\_LINE\_STRIP, GL\_LINES και GL\_POINTS της εντολής glDrawArrays.

* GL\_LINE\_STRIP

Το αποτέλεσμα αυτής της περίπτωσης είναι:

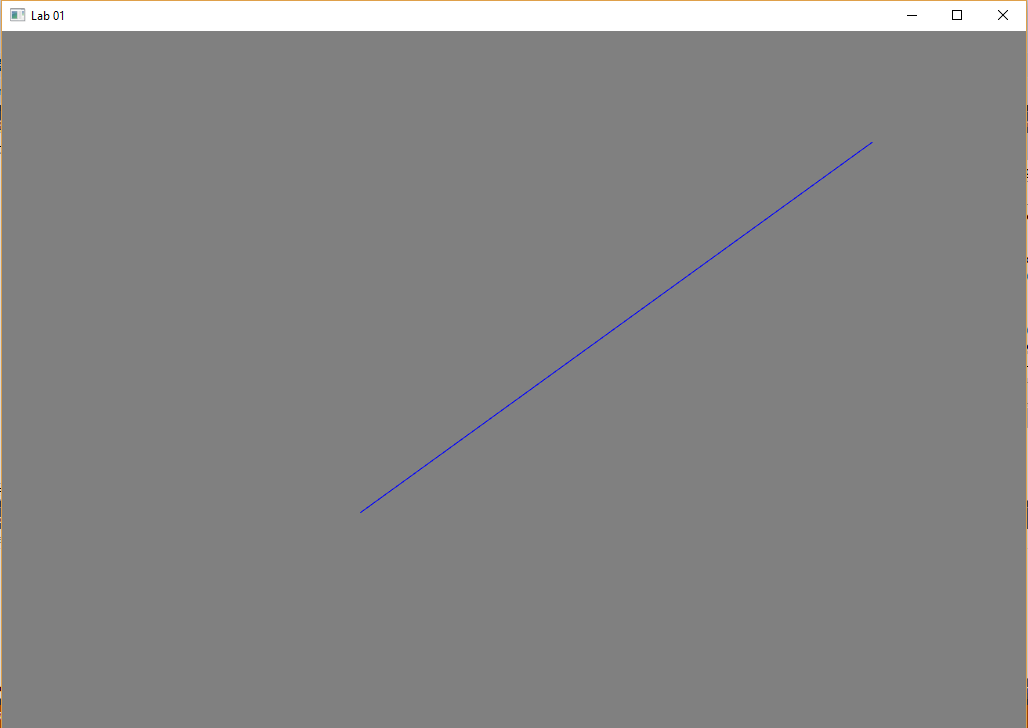


Αυτό συμβαίνει διότι με αυτή η εντολή σχεδιάζει γραμμές από σημείο σε σημείο έτσι στην περίπτωσή μας έδωσα πρώτα το σημείο κάτω αριστερά μετά πάνω δεξιά και τέλος κάτω δεξιά, έτσι σχηματίστηκε το άνωθεν σχήμα αν είχα δώσει τα σημεία με διαφορετική σειρά τότε θα είχα και διαφορετικά αποτελέσματα.



* GL\_LINES

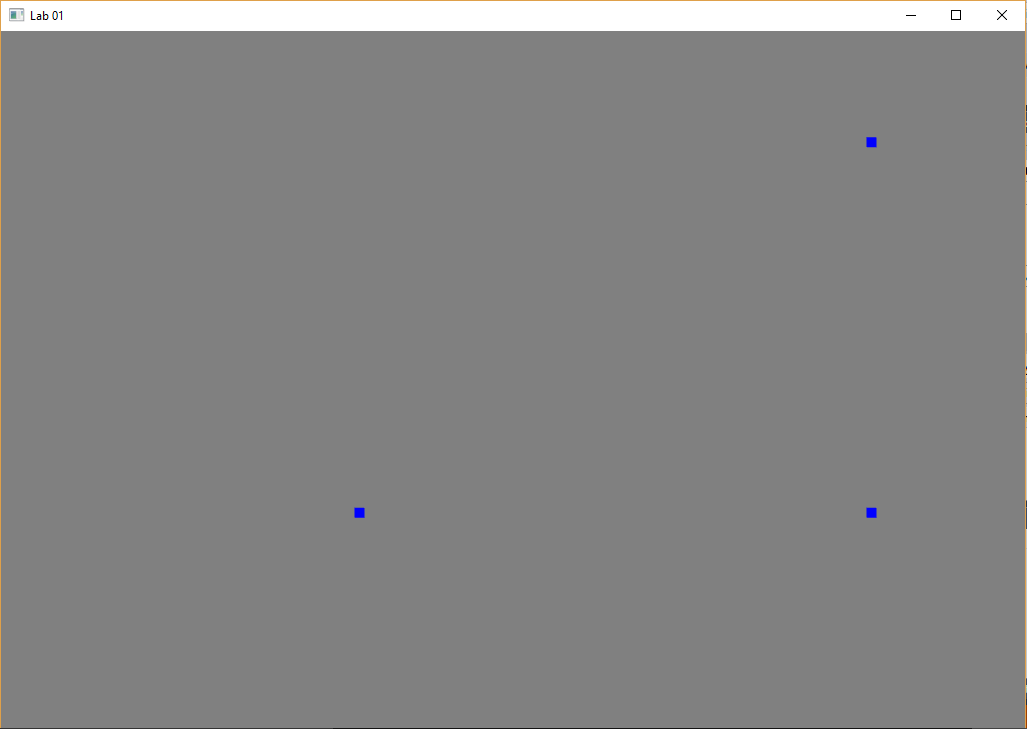
Το αποτέλεσμα αυτής της περίπτωσης είναι:



Αυτό συμβαίνει διότι αυτή η εντολή σχεδιάζει ευθείες αν δυάδες σημείων και εφόσον έχουμε μόνο 2 σημεία δεν μπορεί να σχεδιάσει 2 ευθείες. Επίσης τα σημεία που επιλέγει είναι εκείνα που συναντάει πρώτα, όπως συνέβη και στην πάνω περίπτωση.

* GL\_POINTS

Σε αυτήν την περίπτωση η εντολή σχεδιάζει όλα τα σημεία που έχουν δοθεί . Δεν σχεδιάζουμε την default εντολή, διότι μόνο 1 pixel αντιστοιχεί σε κάθε σημείο οπότε είναι πολύ δυσδιάκριτο. Το αποτέλεσμα του κώδικα αφού ορίσουμε το μέγεθος κάθε σημείου σε 10 (gl\_PointSize = 10;) είναι :



Task 4)

Στόχος αυτού του ερωτήματος είναι να σχεδιάσουμε ένα κύβο. Αυτό το επιτυγχάνουμε σχεδιάζοντας τρίγωνα στις κατάλληλες θέσεις, συγκεκριμένα για αυτή την δουλειά απαιτούνται 12 τρίγωνα. Τα οποία σχεδιάζουμε απλά συμπληρώνοντας στο πίνακα vertices και colors τα στοιχεία των ακμών, καθώς επίσης και αυξάνοντας τα μεγέθη των buffers:

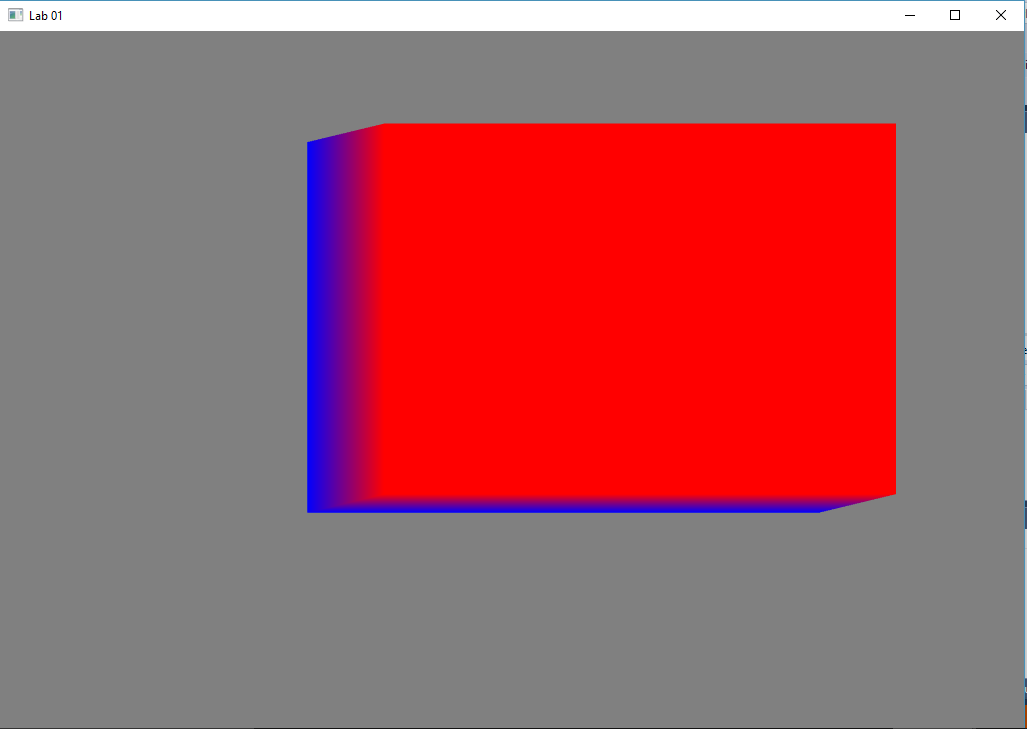
glBufferData(GL\_ARRAY\_BUFFER, NUM\_TRIANGLES \* 9 \* sizeof(GLfloat), vertices, GL\_STATIC\_DRAW);

glBufferData(GL\_ARRAY\_BUFFER, NUM\_TRIANGLES \* 9 \* sizeof(GLfloat), colors, GL\_STATIC\_DRAW);

και ενημερώνοντας την εντολή glDrawArrays για τον αριθμό των τριγώνων που θα σχεδιάσουμε

glDrawArrays(GL\_TRIANGLES, 0, NUM\_TRIANGLES \* 3);

Οπότε ύστερα την συμπλήρωση των παραπάνω έχουμε το παρακάτω αποτέλεσμα:



Το κόκκινο χρώμα δείχνει τα σημεία κοντά μας ενώ το μπλε τα πίσω σημεία.

Μια καλύτερη αίσθηση του βάθους μπορεί να δοθεί χρησιμοποιώντας την εντολή

glPolygonMode(GL\_FRONT\_AND\_BACK, GL\_LINE); αντί για

glPolygonMode(GL\_FRONT\_AND\_BACK, GL\_FILL);

