Εργασία 1

στο μάθημα "Γραφικά με Υπολογιστές"

Καταληκτική ημερομηνία παράδοσης : **Παρασκευή 02 Απριλίου 2021, ώρα 23:55**

Άσκηση 1

Για κάθε Αριθμό Μητρώου (ΑΜ) φοιτητή/τρια θα υπολογιστεί ένας αριθμός ο οποίος καθορίζει τις διαφορετικές αναθέσεις. Ο αριθμός αυτός προκύπτει από τον τύπο :

U = AM mod 6

Ανάλογα με το **U** που έχει ο κάθε φοιτητής/τρια θα πρέπει να σχεδιαστούν δύο διαφορετικά πολύγωνα όπως φαίνεται παρακάτω:

Για U=0 θα σχεδιαστούν τα πολύγωνα με ID : 0,2
Για U=1 θα σχεδιαστούν τα πολύγωνα με ID : 1,3
Για U=2 θα σχεδιαστούν τα πολύγωνα με ID : 2,4
Για U=3 θα σχεδιαστούν τα πολύγωνα με ID : 3,5
Για U=4 θα σχεδιαστούν τα πολύγωνα με ID : 4,0
Για U=5 θα σχεδιαστούν τα πολύγωνα με ID : 5,1

Αντιστοιχία αριθμών και τύπου πολυγώνου	
ID	**Τύπος Πολυγώνου**
0	πεντάγωνο
1	εξάγωνο
2	επτάγωνο
3	οκτάγωνο
4	εννεάγωνο
5	δεκάγωνο

Να δημιουργηθεί αρχείο κώδικα (.cpp) με shaders κατά το οποίο θα παράγεται ένα παράθυρο OpenGL στο οποίο θα εμφανίζονται τα εξής:

Τα δύο διαφορετικά πολύγωνα, με διαφορετικό αντικείμενο VBO (Vertex Buffer Object) για κάθε πολύγωνο, τοποθετημένα σε διαφορετικές θέσεις και με διαφορετικά χρώματα. Να ορισθεί μια μεταβλητή uniform float με όνομα intensity, η οποία θα έχει εύρος [0.0, 1.0] και θα πολλαπλασιάζεται με το χρώμα κάθε σχήματος. Αυτή η μεταβλητή θα μπορεί να αυξάνεται με το πλήκτρο "Arrow Up" και να μειώνεται με το πλήκτρο "Arrow Down".

Επιπλέον, θα είναι δυνατή η εναλλαγή μεταξύ του "solid mode" και του "wireframe mode" με τη χρήση του πλήκτρου 'W'.

Υποσημείωση 1: Για να γίνεται η αλλαγή με κάθε πάτημα του πλήκτρου μόνο μία φορά και όχι συνεχόμενα, θα χρειαστεί να ορισθούν δύο boolean μεταβλητές, οι οποίες θα σχετίζονται με το state του πλήκτρου στο τρέχον και το προηγούμενο frame.

Υποσημείωση 2: Για την κατασκευή ενός πολυγώνου να χρησιμοποιηθεί το mode **GL_POLYGON** αντί για το GL_TRIANGLES. Το mode **GL_POLYGON** συνδέει όλες τις κορυφές με την σειρά όπως έχουν οριστεί στο πίνακα με τις κορυφές και επίσης συνδέει και την τελευταία με την πρώτη κορυφή.

Το αρχείο κώδικα της άσκησης 1 θα έχει όνομα "E1_A1_{AM}.cpp", όπου {AM} θα είναι ο αριθμός μητρώου σας. Ο vertex shader θα έχει όνομα "E1_A1_VertexShader.txt" και ο fragment shader θα έχει όνομα "E1_A1_FragmentShader.txt".

Άσκηση 2

Να δημιουργηθεί αρχείο κώδικα (.cpp) με shaders κατά το οποίο θα παράγεται ένα παράθυρο OpenGL στο οποίο θα εμφανίζονται τα εξής:

Ένα πολύγωνο (να επιλέξετε ένα από τα δύο πολύγωνα που έχουν παραχθεί στην άσκηση 1), το οποίο θα έχει τη δυνατότητα να κινείται στην οθόνη με το πάτημα πλήκτρων από τον χρήστη. Για την κίνηση προς τα πάνω (θετικός άξονας Y) θα χρησιμοποιείται το πλήκτρο "W", για την κίνηση προς τα κάτω (αρνητικός άξονας Y) το πλήκτρο "S", για την κίνηση προς τα δεξιά (θετικός άξονας X) το πλήκτρο "D", και για την κίνηση προς τα αριστερά (αρνητικός άξονας X) το πλήκτρο "A". Το πολύγωνο θα μπορεί να τοποθετείται οπουδήποτε στην οθόνη, χωρίς να μετακινείται έξω από τα όριά της.

Το χρώμα (R,G,B) κάθε κορυφής του πολυγώνου (x, y, z), με $-1.0 \leq x, y, z \leq 1.0$ θα προκύπτει από την θέση της κορυφής, αλλά με την κατάλληλη κανονικοποίηση ώστε να τηρούνται οι επόμενες προϋποθέσεις:

- Η τιμή 'R' θα παίρνει τιμή 0.0 στις ακραίες αριστερά θέσεις της οθόνης για x = -1.0, τιμή 1.0 στις ακραίες δεξιά θέσεις της οθόνης για x = 1.0 και ενδιάμεσες τιμές σε όλες τις άλλες θέσεις.
- Η τιμή 'G' θα παίρνει τιμή 0.0 στις ακραίες κάτω θέσεις της οθόνης για y = -1.0, τιμή 1.0 στις ακραίες πάνω θέσεις της οθόνης για y = 1.0 και ενδιάμεσες τιμές σε όλες τις άλλες θέσεις.
- Η τιμή 'B' θα είναι πάντα μηδενική.

Το αρχείο κώδικα της άσκησης 2 θα έχει όνομα "E1_A2_{AM}.cpp", όπου {AM} θα είναι ο αριθμός μητρώου σας. Ο vertex shader θα έχει όνομα "E1_A2_VertexShader.txt" και ο fragment shader θα έχει όνομα "E1_A2_FragmentShader.txt".

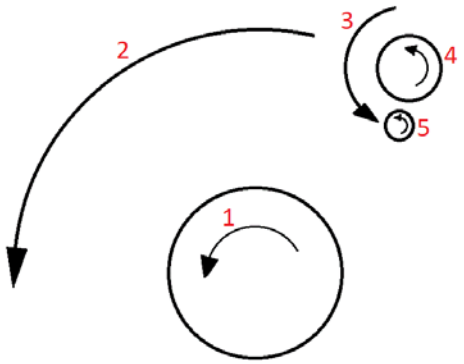
Άσκηση 3

Να δημιουργηθεί αρχείο κώδικα (.cpp) με shaders κατά το οποίο θα παράγεται ένα παράθυρο OpenGL στο οποίο θα εμφανίζονται τα εξής:

Τρεις κύβοι, οι οποίοι θα κινούνται στο χρόνο.

- Ο **πρώτος κύβος**, θα είναι στο κέντρο της οθόνης και θα περιστρέφεται γύρω από τον εαυτό του (περιστροφή 1).
- Ο **δεύτερος κύβος** θα περιστρέφεται γύρω από τον πρώτο (περιστροφή 2) και γύρω από τον εαυτό του (περιστροφή 4).
- Ο **τρίτος κύβος**, θα περιστρέφεται γύρω από τον πρώτο κύβο (περιστροφή 2), γύρω από τον δεύτερο (περιστροφή 3) και γύρω από τον εαυτό του (περιστροφή 5).

Οι περιστροφές των κύβων φαίνονται στο παρακάτω σχήμα:



Οι περιστροφές 2 και 3 να είναι γύρω από τον άξονα (1, 1, 0). Οι περιστροφές 1, 4 και 5 ναι είναι γύρω από διαφορετικούς άξονες. Να χρησιμοποιηθεί το μοντέλο του κύβου με διαφορετικό χρώμα σε κάθε πλευρά, το οποίο έχουμε χρησιμοποιήσει και στο εργαστήριο.

Το μέγεθος του κάθε κύβου θα καθορίζεται από τα 3 τελευταία ψηφία του ΑΜ. Το τρίτο ψηφίο θα καθορίζει το μέγεθος του πρώτου κύβου, το τέταρτο του δεύτερου κύβου και το πέμπτο του τρίτου κύβου. Τα ψηφία αυτά θα αντιπροσωπεύουν το ποσοστό αποκλιμάκωσης για όλους τους άξονες για κάθε διαφορετικό κύβο. Για παράδειγμα για ΑΜ = 56049, στον πρώτο κύβο θα εφαρμοστεί αποκλιμάκωση κατά 1.0 σε όλους τους άξονες, στον δεύτερο κύβο κατά 1.4 σε όλους τους άξονες και στον τρίτο κύβο κατά 1.9 σε όλους του άξονες.

Η κάμερα θα πρέπει να έχει στο οπτικό της πεδίο και τους τρεις κύβους .

Το αρχείο κώδικα της άσκησης 3 θα έχει όνομα "E1 _A3_{AM}.cpp", όπου {AM} θα είναι ο αριθμός μητρώου σας. Ο vertex shader θα έχει όνομα "E1_A3_VertexShader.txt" και ο fragment shader θα έχει όνομα "E1_A3_FragmentShader.txt".

Οι shaders θα πρέπει να τοποθετηθούν σε ένα φάκελο με όνομα "Shaders_{AM}", όπου {AM} ο αριθμός μητρώου σας. Το αρχείο κώδικα και ο φάκελος των shaders να τοποθετηθούν σε έναν φάκελο με όνομα "Ergasia1_{AM}", ο οποίος θα συμπιεσθεί σε αρχείο (.rar, .zip) και θα υποβληθεί στο Eclass.

Φροντίστε ώστε τα ονόματα των path των shaders να έχουν την μορφή “res/Shaders_{AM}/ E1_A3_VertexShader.txt ” ώστε ο κώδικας να τρέχει απευθείας όταν φορτώσουμε το αρχείο κώδικα της κάθε άσκησης και το φάκελο "Shaders_{AM}" στο project που έχουμε δημιουργήσει.