**ΠΡΟΓΡΑΜΜΑΤΙΣΤΙΚΗ ΑΣΚΗΣΗ 1-A (2023)**

Γραφικά Υπολογιστών και Συστήματα Αλληλεπίδρασης

Στοιχεία ομάδας:

 Όνομα: Δημοσθένης Ζάγκας ΑΜ: 4359

  Όνομα: Ανδρέου Άγγελος ΑΜ: 4628

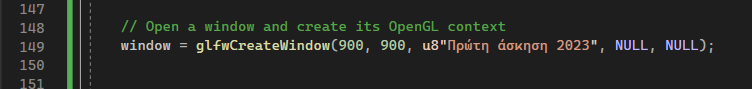
Ημερομηνία : 03/11/2023



1. **ΕΙΣΑΓΩΓΗ**

Στα πλαίσια του εργαστηρίου του μαθήματος “Γραφικά Υπολογιστών και Συστήματα Αλληλεπίδρασης” μας ζητήθηκε να κάνουμε μία εισαγωγική άσκηση OpenGL (1-Α) ώστε να εξοικειωθούμε με τα βασικά στοιχεία και τις δυνατότητες που προσφέρει. Στο πρόγραμμα δημιουργήσαμε ένα παράθυρο που εμφανίζεται ένα τετράγωνο να μεταφέρεται σε 5 θέσεις. Το βασικό αρχείο που δουλέψαμε το πρόγραμμα κάνοντας αλλαγές πάνω σε αυτό είναι το “Source-1A.cpp” που μας δόθηκε με το υλικό για την άσκηση.

1. **Περιγραφή της εργασίας**
2. Αρχικά για να ανοίγει το παράθυρο σε 900x900 διαστάσεις και με το όνομα που ζητήθηκε αλλάξαμε τη συνάρτηση της glfw ως εξής:

window = glfwCreateWindow(900, 900, u8"Πρώτη άσκηση 2023", NULL, NULL);

1. Στο δεύτερο ζητούμενο καθορίζουμε τα vertices για τα τετράγωνα:

 Αρχικά πρέπει να βρούμε τα 4 σημεία στα επίπεδα που εμφανίζεται το κάθε τετράγωνο με σειρά. Αφού έχουμε 2-Δ σχήμα η συνιστώσα στον άξονας z είναι πάντα 0 οπότε θα παραλείπεται στην εξήγηση. Αφού η πλευρά το τετραγώνου είναι 3 και το κέντρο του η αρχή των αξόνων έχουμε τα σημεία (εικόνα 1):

A piece of paper with writing on it

Description automatically generatedεικόνα 1

Συμμετρικά θέλουμε 1.5 μονάδες απόσταση από το (0,0) ώστε η συνολική τους απόσταση να είναι 3.

Για τα υπόλοιπα τετράγωνα δουλεύουμε αντίστοιχα προσέχοντας το 1 κοινό σημείο που έχουν με το αρχικό τετράγωνο (εικόνα 2).

A white paper with writing on it

Description automatically generated

 εικόνα 2

Στην OpenGL πρέπει να σχηματίσουμε τα τετράγωνα σαν 2 τρίγωνα με μία κοινή πλευρά και 2 κοινές κορυφές. Για παράδειγμα για το πρώτο τετράγωνο (εικόνα 1) σημειώνουμε τα 2\*3=6 vertices ως εξής:

// First triangle

// Points: a=(1.5,1.5,0), b=(-1.5,-1.5,0), c=(1.5,-1.5,0)

// x-distance b to c is from -1.5 to 1.5 = 3, as per requirements

// y-distance a to b is from -1.5 to 1.5 = 3, as per requirements

1.5f,1.5f,0.0f,

1.5f,-1.5f,0.0f,

-1.5f,-1.5f,0.0f,

// Second triangle

1.5f,1.5f,0.0f,

-1.5f,1.5f,0.0f,

-1.5f,-1.5f,0.0f,

Για τα υπόλοιπα τετράγωνα:

// Second square

// First triangle

1.5f,1.5f,0.0f,

4.5f,1.5f,0.0f,

4.5f,4.5f,0.0f,

// Second triangle

1.5f,1.5f,0.0f,

1.5f,4.5f,0.0f,

4.5f,4.5f,0.0f,

// Third square

// First triangle

-1.5f,-1.5f,0.0f,

-1.5f,-4.5f,0.0f,

-4.5f,-4.5f,0.0f,

// Second triangle

-1.5f,-1.5f,0.0f,

-4.5f,-1.5f,0.0f,

-4.5f,-4.5f,0.0f,

// Forth square

// First triangle

-1.5f,1.5f,0.0f,

-1.5f,4.5f,0.0f,

-4.5f,4.5f,0.0,

// Second triangle

-1.5f,1.5f,0.0f,

-4.5f,1.5f,0.0f,

-4.5f,4.5f,0.0f,

// Fifth square

// First triangle

1.5f,-1.5f,0.0f,

1.5f,-4.5f,0.0f,

4.5f,-4.5f,0.0f,

// Second triangle

1.5f,-1.5f,0.0f,

4.5f,-1.5f,0.0f,

4.5f,-4.5f,0.0f

Τα οποία αποθηκεύονται στον πίνακα:

static const GLfloat shape\_1\_buffer[] {...}

A screenshot of a computer program

Description automatically generatedA computer screen with green and white text

Description automatically generated

1. Για το iii) ερώτημα θα γίνεται εμφάνιση μέσα σε επανάληψη do while όπως στο πρότυπο και το test που είδαμε στο εργαστήριο, με τις διαφοροποιήσεις:

Χρήση των παραμέτρων:

int square\_number = 0;

//για να ξέρουμε το τετράγωνο που πρέπει να εμφανιστεί σε κάθε φάση, αυξάνεται μετά από την εμφάνιση του μεσαίου, αρχικού τετραγώνου

bool flag = true;

//flag για έλεγχο αν εμφανίζουμε το μεσαίο τετράγωνο sq1->sq2->!sq1!->sq3 έτσι δεν χρειάζεται να επαναλάβουμε vertices στον πίνακα buffer

double wait\_time = 1;//secs

//χρόνος αναμονής που θα γίνεται εμφάνιση κάθε τετραγώνου πριν πάμε στο επόμενο

double time = 0;//secs

//πόσο χρόνο έχουμε στην εμφάνιση αυτού του τετραγώνου

Για να υπολογίσουμε τον χρόνο που εμφανίζεται το ίδιο τετράγωνο κάναμε χρήση των time συναρτήσεων της glfw:

 //get the time for transition

<https://www.glfw.org/docs/3.3/group__input.html#gaa6cf4e7a77158a3b8fd00328b1720a4a>

glfwSetTime(0);

time = glfwGetTime();//secs

//wait time for transition

Έτσι γίνεται συνεχόμμενα εμφάνιση του ενός τετραγώνου μέχρι να φτάσει το wait\_time ή τερματίσουμε το πρόγραμμα

//check time for wait

time = glfwGetTime() - time;

} while (wait\_time > time && glfwGetKey(window, GLFW\_KEY\_C) != GLFW\_PRESS && glfwWindowShouldClose(window) == 0);

Για να εμφανίσουμε τα τρίγωνα μέσα στη do-while έχουμε

// Draw the triangle !

if (flag) //we are at 1st squad

glDrawArrays(GL\_TRIANGLES, 0, 6); // draw square 1

else

glDrawArrays(GL\_TRIANGLES, square\_number, 6);

// 6 indices starting at square\_number -> 2 triangles

Μετά αφού εμφανιστεί το τετράγωνο για το χρόνο που ορίσαμε έξω από το πρώτο do-while αλλάζουμε το square\_number ώστε να πάμε στο επόμενο (το αρχικό αν ήμασταν σε κάποιο από τα εξωτερικά τετράγωνα ή το επόμενο μετά το κεντρικό σύμφωνα με τη σειρά που μας ζητήθηκε)

//if done 1st square go next

flag = !flag;

// Increase the square number after every draw to render the next set of vertices that create the next square

if (square\_number <= 24 && !flag) {

square\_number += 6;

}

// When the last square is drawn, start over

if ( square\_number > 24) {

square\_number = 0;

flag = true;

}

Έτσι θα έχουμε διαδοχικά τις εκτελέσεις:

glDrawArrays(GL\_TRIANGLES, 0, 6);

glDrawArrays(GL\_TRIANGLES, 6, 6);

glDrawArrays(GL\_TRIANGLES, 0, 6);

glDrawArrays(GL\_TRIANGLES, 12, 6);

glDrawArrays(GL\_TRIANGLES, 0, 6);

glDrawArrays(GL\_TRIANGLES, 18, 6);

glDrawArrays(GL\_TRIANGLES, 0, 6);

glDrawArrays(GL\_TRIANGLES, 24, 6);

Μέχρι να πατηθεί το κουμπί c ή κλείσουμε το παράθυρο:

while (glfwGetKey(window, GLFW\_KEY\_C) != GLFW\_PRESS && glfwWindowShouldClose(window) == 0);

A computer screen with text and numbers

Description automatically generated

A computer code on a black background

Description automatically generated

A computer screen with text

Description automatically generated

1. Στο 4ο ερώτημα για αλλαγή του χρόνου μετάβασης με τα κουμπιά u , d:

Ελέγχουμε μέσα στο πρώτο do-while αν πατηθεί κάποιο από τα δύο κουμπιά. Αν πατηθεί το “u” θέλουμε να αυξάνεται ο ρυθμός έτσι μειώνουμε το χρόνο που εμφανίζεται το τετράγωνο (wait\_time ) κατά 0.7.

Αν πατηθεί το “d” θέλουμε να μειώνεται ο ρυθμός έτσι αυξάνουμε το χρόνο που εμφανίζεται το τετράγωνο (wait\_time ) κατά 1.2.

//change transition time with u key

if (glfwGetKey(window, GLFW\_KEY\_U)) {

wait\_time = wait\_time \* 0.7;

}

if (glfwGetKey(window, GLFW\_KEY\_D)) {

wait\_time = wait\_time \* 1.2;

}

* Όλα τα ερωτήματα υλοποιήθηκαν.

1. **Πληροφορίες σχετικά με την υλοποίηση:**

* Λειτουργικό σύστημα: Windows 64-bit
* Περιβάλλον: Visual Studio versions 17.7.6
* Παρότι το λειτουργικό σύστημα είναι 64-bit μπορεί χωρίς πρόβλημα να τρέξει εφαρμογές 32-bit
* Η συγγραφή αυτού του report έγινε σε google docs.

1. **Σύντομη αξιολόγηση της λειτουργίας της ομάδας:**

Η ομάδα λειτούργησε χωρίς προβλήματα κυρίως εξ αποστάσεως με χρήση του gitHub για συντονισμό αρχείων. Μαζί κάναμε τον σχεδιασμό που απαιτήθηκε, με τον Δημοσθένη να κάνει την εγγραφη του κώδικα και το πέρασμα των vertices των τετραγώνων ενώ ο Άγγελος ασχολήθηκε με τη σωστή λειτουργία και τον ρυθμό εναλλαγής του μοτίβου και με τη συγγραφή αυτής της αναφοράς.

1. **Αναφορές:**

* <https://www.glfw.org/docs/3.3/group__input.html#gaa6cf4e7a77158a3b8fd00328b1720a4a>
* <https://www.glfw.org/docs/3.3/quick_guide.html>
* <https://www.glfw.org/docs/3.3/window_guide.html>
* <https://www.opengl-tutorial.org/>
* https://www.wikihow.com/Set-Up-OpenGL-GLFW-GLEW-GLM-on-a-Project-with-Visual-Studio