ΠΑΝΕΠΙΣΤΗΜΙΟ ΙΩΑΝΝΙΝΩΝ

ΤΜΗΜΑ ΜΗΧ. Η/Υ & ΠΛΗΡΟΦΟΡΙΚΗΣ

Γραφικά Υπολογιστών και Συστήματα Αλληλεπίδρασης

Β Μέρος -Σύνολο Προγραμματιστικών Ασκήσεων

Μέλη ομάδας :

Μανίκας Ελευθέριος Μάριος ,4723

Φωτόπουλος Στέφανος ,4829

Ιωάννινα,18/11/2022

Contents

1.Περιγραφή της εργασίας

Ερώτημα 1

Σε αυτήν την γραμμή δημιουργούμε ένα καινούργιο παράθυρο, συγκεκριμένα τετράγωνο με διαστάσεις 1000*1000 και όνομα **«Εργασία 1Β – Τραπεζοειδές Πρίσμα».** Επίσης προσθέσαμε το u8 για να εμφανίζονται σωστά τα ελληνικά στο window (κωδικοποίηση).

```
// Open a window and create its OpenGL context
window = glfwCreateWindow(1000, 1000, u8"Εργασία 1Β - Τραπεζοειδές Πρίσμα", NULL, NULL);
```

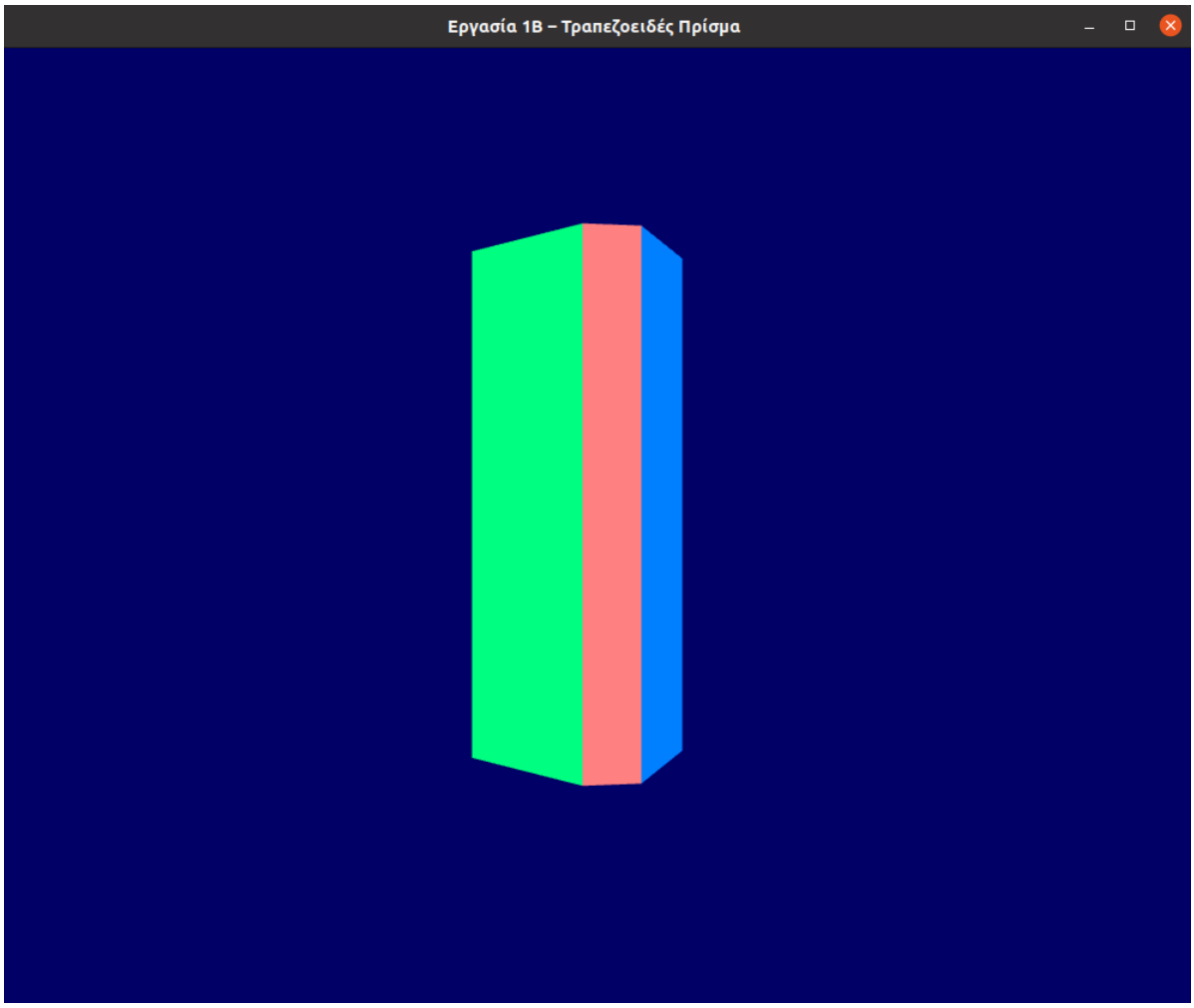
Αυτή η εντολή χρησιμεύει στον καθορισμό του χρώματος του background ,οι 3 πρώτες τιμές αφορούν η κάθε μια με την σειρά το red,green,blue(rgb). Έτσι αυξήσαμε το blue σε κάποια τιμή για να πετύχουμε μια dark απόχρωση του μπλε και συγκεκριμένα στο 0.4.

```
//background color
glClearColor(0.0f, 0.0f, 0.4f, 0.0f);//0.4 dark blue
```

Μέσα σε αυτήν την του while γίνεται ο έλεγχος για τον τερματισμό του προγράμματος. Χρησιμοποιήσαμε την εντολή GLFW_KEY_SPACE, έτσι ώστε ο τερματισμός να γίνεται αν πατήσουμε space όταν είναι ενεργό το παράθυρο η αν κλείσει από το X.

```
// Check if the SPACE key was pressed or the window was closed
while (glfwGetKey(window, GLFW_KEY_SPACE) != GLFW_PRESS &&
       glfwWindowShouldClose(window) == 0);
```

Ερώτημα 2



Υπολογισμός σημείων πρίσματος.

Εφόσον θέλουμε το κέντρο του πρίσματος να είναι το P(0,0,0), δηλαδή η αρχή των αξόνων .

Η άνω βάση μας θα είναι κατά h/2 αυξημένη στον άξονα z, πάνω από την αρχή των αξόνων και με κέντρο το P(0,0, h/2).Για να βρούμε τις συντεταγμένες των κορυφών της άνω βάσης ακολουθήσαμε την παρακάτω διαδικασία. Αρχικά c=6 , άρα θέλουμε να μεταφερθούμε 3 μονάδες πάνω και 3 κάτω από την αρχή των αξόνων .Οπότε η πάνω πλευρά του τραπεζίου θα εχει y=3 και η κάτω πλευρά του τραπεζίου y=-3.Οσο για το X δίνεται ότι α=2 για την μικρή πλευρά του τραπεζίου, το κέντρο της είναι στο x=0, άρα η αριστερή κορυφή θα εχει x=-1 (-1,3,h/2) και η δεξιά κορυφή x=1 (1,3,h/2).Επειδή η κάτω πλευρά του τραπεζίου μας είναι 8 ομοίως θα έχουμε x=-4 (-4,-3,h/2) για την κάτω αριστερή κορυφή του τραπεζίου και x=4 (4,-3,h/2) για την κάτω δεξιά πλευρά του τραπεζίου μας.

Η κάτω βάση έχεις όλες τις τιμές της για το x και το y ίδιες με την άνω βάση με μόνη εξαίρεση τον άξονα Z στον οποίο όλες οι κορυφές της έχουν τιμή -h/2.

Το μπροστινό μικρότερο ορθογώνιο εχει τις πάνω κορυφές του ίδιες με τις κορυφές της μικρής πλευράς της άνω βάσης και τις κάτω κορυφές του ίδιες με τις κορυφές της μικρής πλευράς της κάτω βάσης.

Ομοίως υπολογίζουμε και τις κορυφές των άλλων τριών ορθογωνίων και τελικά προκύπτουν οι εξής συντεταγμένες :

```
268         //syntetagmenes koryfwn prismatos
269         static const GLfloat g_vertex_buffer_data[] = {
270
271                 //anw vasi me z=h/2
272                 -1.0f,3.0f,h / 2,
273                 1.0f,3.0f,h / 2,
274                 4.0f,-3.0f,h / 2,
275
276                 -1.0f,3.0f,h / 2,
277                 -4.0f,-3.0f,h / 2,
278                 4.0f,-3.0f,h / 2,
279                 //katw vasi me z=-h/2
280                 -1.0f,3.0f,-h / 2,
281                 1.0f,3.0f,-h / 2,
282                 4.0f,-3.0f,-h / 2,
283
284                 -1.0f,3.0f,-h / 2,
285                 -4.0f,-3.0f,-h / 2,
286                 4.0f,-3.0f,-h / 2,
287
288                 //messaio orthogwnio
289                  -4.0f,-3.0f, h / 2,// h/2
290                  4.0f,-3.0f, h / 2,// h/2
291                  4.0f,-3.0f, -h / 2,// -h/2
292
293                  -4.0f,-3.0f, h / 2,// h/2
294                  4.0f,-3.0f,-h / 2,// -h/2
295                  -4.0f,-3.0f, -h / 2,// -h/2
296
297                  //aristero orthogwnio
298                  -4.0f,-3.0f, h / 2,// h/2
299                  -1.0f,3.0f, h / 2,// h/2
300                  -4.0f,-3.0f, -h / 2,// -h/2
301
302                  -1.0f,3.0f, h / 2,// h/2
303                  -1.0f,3.0f, -h / 2,// -h/2
304                  -4.0f,-3.0f, -h / 2,// -h/2
305                  //deksi orthogwnio
306                   4.0f,-3.0f, h / 2,// h/2
307                   1.0f,3.0f, -h / 2,// -h/2
308                   4.0f,-3.0f, -h / 2,// -h/2
309
310                   1.0f,3.0f, h / 2,// h/2
311                   1.0f,3.0f, -h / 2,// -h/2
312                   4.0f,-3.0f, h / 2,// h/2
```

```
313                 //mprostino orthogwnio
314                 -1.0f,3.0f, h / 2,// h/2
315                 1.0f,3.0f, h / 2,// h/2
316                 1.0f,3.0f, -h / 2,// -h/2
317
318                 -1.0f,3.0f, h / 2,// h/2
319                 -1.0f,3.0f,-h / 2,// -h/2
320                 1.0f,3.0f, -h / 2,// -h/2
321
```

Το πρίσμα μας αποτελείται από 4 ορθογώνια και 2 τραπέζια τα οποία σχηματίζονται από δυο τρίγωνα το καθένα, άρα συνολικά 12 τρίγωνα.

Ο χρωματισμός κάθε σχήματος ξεχωριστά γίνεται αν βάλουμε την ίδια χρωματική απόχρωση σε κάθε κορυφή των δυο τρίγωνων που αποτελούν το τραπέζιο ή το ορθογώνιο μας και υλοποιείται στις γραμμές κώδικα 324 με 374.

Το ύψος του πρίσματος $\boldsymbol{h}$ προσδιορίζεται τυχαία με γεννήτρια τυχαίων αριθμών στο διάστημα [2.0, 10.0].Έτσι χρησιμοποιήσαμε την συνάρτηση srand όπως φαίνεται παρακάτω :

```
float h = 0;
float min = 2.0;
float max = 10.0;
srand((unsigned)time(NULL));
h = (min)+(((float)rand()) / (float)RAND_MAX) * (max - min);
cout << h;
```

Σημείωση: Για δική μας βοήθεια και να βλέπουμε πως αλλάζει ο κάθε πίνακας σε διαφορετικές χρονικές στιγμές χρησιμοποιήσαμε τον παρακάτω κώδικα ο οποίος εκτυπώνει τα περιεχόμενα ενός 4*4 πίνακα (mat4).

```
for (int i = 0; i < 4; i++) {
    for (int j = 0; j < 4; j++) {
      cout << PinakasEktiposis[j][i];
    }
  }
```

Ερώτημα 3

```
//********************
// **Δείγμα κλήσης των συναρτήσεων για δημιουργία του MVP - είναι τυχαίες οι ρυθμίσεις και δεν ανταποκρίνονται στην άσκησή σας

glm::mat4 Projection = glm::perspective(glm::radians(45.0f), float(1000 / 1000), 0.1f, 100.0f);
// Camera matrix
glm::mat4 View = glm::lookAt(
        glm::vec3(10.0f, 50.0f, 0.0f),//topothesia kameras erotimatos
        glm::vec3(0.0f, 0.0f, 0.0f),// kentro
        glm::vec3(0.0f, 0.0f, 1.0f)//anion dianysma
);
// Model matrix : an identity matrix (model will be at the origin)
glm::mat4 Model = glm::mat4(1.0f);
// Our ModelViewProjection : multiplication of our 3 matrices
glm::mat4 MVP = Projection * View * Model; // Remember, matrix multiplication is the other way around
ViewMatrix = View;//thetw to ViewMatrix = View gia na mporw na xrhsimopoihsw thn getViewMatrix() sthn camera_function()
ProjectionMatrix = Projection;//thetw to ProjectionMatrix = Projection gia na mporw na xrhsimopoihsw thn getProjectionMatrix() sthn camera_function()
```

Για να υλοποιήσουμε το ερώτημα της άσκησης χρειάζεται να ορίσουμε τον πίνακα View με την συνάρτηση lookAt η οποία ελέγχει τον τρόπο λειτουργίας της κάμερας. Συγκεκριμένα το πρώτο vector είναι η τοποθέτηση της κάμερας στον πραγματικό κόσμο(10.0, 50.0, 0.0) και το δεύτερο vector είναι για το σημείο προς το οποίο θα κοιτάει η κάμερα P(0,0,0) (αρχή των αξόνων).Επίσης προσθέσαμε τις δυο τελευταίες εντολές έτσι ώστε να μπορούμε να χρησιμοποιήσουμε την getViewMatrix() και την getProjectionMatrix() στην camera_function() που δεν είναι στην εμβέλεια της main().

Ερώτημα 4

```
//Scaling
if (glfwGetKey(window, GLFW_KEY_U) == GLFW_PRESS)//elegxos gia to patima koumpiou
{
        Model = glm::scale(Model, glm::vec3(1.0f, 1.0f, 1.2f));//kanw scale ton monadiaio pinaka Model me to vector 3(x,y,z)
}
if (glfwGetKey(window, GLFW_KEY_P) == GLFW_PRESS)//elegxos gia to patima koumpiou
{
        Model = glm::scale(Model, glm::vec3(1.0f, 1.0f, 0.8f));//kanw scale ton monadiaio pinaka Model me to vector 3(x,y,z)
}
```

Ο παραπάνω κώδικας απεικονίζει την λειτουργία του scaling. Αρχικά ελέγχουμε για το πάτημα του πλήκτρου U ή του πλήκτρου P. Αν πατηθεί το πλήκτρο U πειράζουμε τον πίνακα model ο οποίος στην αρχή είναι μοναδιαίος , δηλαδή τον κάνουμε scale με ένα vector ο οποίος αποτυπώνει μέσα το (x,y,z), επειδή θέλουμε να αλλάξουμε μόνο το h το οποίο είναι στον άξονα z αυξάνουμε μόνο τη τελευταία συντεταγμένη (η οποία προκαλεί αλλαγή στο h).Αντίστοιχα όταν πατηθεί το πλήκτρο P μειώνουμε την τιμή του h για να πετύχουμε το scale down.

```
MVP = Projection * View * Model;// ypologizw to kainourgio mvp symfwna me ta kainourgia dedomena
```

Τέλος υπολογίζουμε το καινούργιο MVP με το καινούργιο Model που προέκυψε από την scale για να εμφανίζεται σωστά το σχήμα μας στην οθόνη.

Ερώτημα 5

Υλοποίηση της κάμερας η οποία ελέγχεται μόνο με τα πλήκτρα του πληκτρολογίου

```
void camera_function()
{
	// STRAFE STON AXONA X  UP
	if (glfwGetKey(window, GLFW_KEY_W) == GLFW_PRESS) {
		ViewMatrix = glm::rotate(ViewMatrix, radians(1.0f), glm::vec3(1.0f, 0.0f, 0.0f));
	}
	//STRAFE STON AXONA X Down
	if (glfwGetKey(window, GLFW_KEY_X) == GLFW_PRESS) {
		ViewMatrix = glm::rotate(ViewMatrix, radians(1.0f), glm::vec3(-1.0f, 0.0f, 0.0f));

	}
	//STRAFE STON AXONA Z  Dexiostrofa
	if (glfwGetKey(window, GLFW_KEY_Q) == GLFW_PRESS) {
		ViewMatrix = glm::rotate(ViewMatrix, radians(1.0f), glm::vec3(0.0f, 0.0f, 1.0f));
	}
	//STRAFE STON AXONA Z Aristerostrofa
	if (glfwGetKey(window, GLFW_KEY_Z) == GLFW_PRESS) {
		ViewMatrix = glm::rotate(ViewMatrix, radians(1.0f), glm::vec3(0.0f, 0.0f, -1.0f));
	}
	// ZOOM IN
	if (glfwGetKey(window, GLFW_KEY_KP_ADD) == GLFW_PRESS) {
		zoom = zoom - 0.1;
		if (zoom >= 0.1)
		{
			ProjectionMatrix = glm::perspective(glm::radians(zoom), float(1000 / 1000), 0.1f, 100.0f);
		}
	}
	//ZOOM OUT
	if (glfwGetKey(window, GLFW_KEY_KP_SUBTRACT) == GLFW_PRESS)
	{
		zoom = zoom + 0.1;
		if (zoom <= 100.0)
		{
			ProjectionMatrix = glm::perspective(glm::radians(zoom), float(1000 / 1000), 0.1f, 100.0f);
		}
	}
}
```

i) Περιστροφή γύρω από τον άξονα x με τα πλήκτρα W και X

Αρχικά γίνεται ο έλεγχος στα if αν εχει πατηθεί το ζητούμενο πλήκτρο. Στην συνέχεια επειδή θελουμε να περιστρέψουμε την κάμερα θα τροποποιήσουμε το ViewMatrix στο οποίο έχουμε εισάγει το περιεχόμενο της View που προέκυψε με την εντολή lookAt.Έτσι χρησιμοποιούμε την εντολή rotate με ορίσματα το ViewMatrix,μια γωνία σε μοίρες και ένα vector που κρατάει τις συντεταγμένες (x,y,z).Με το πλήκτρο W επιλέξαμε να κινείται στον άξονα X με κατεύθυνση προς τα πάνω άρα το vector είναι το (1,0,0) και αντίστοιχα με το πλήκτρο X κινείται στον άξονα X προς τα κάτω με το vector (-1,0,0). Την γωνιά την κρατήσαμε σταθερή σε κάθε rotate στην τιμή radians(1.0f). Το αποτέλεσμα του rotate το αποθηκεύουμε στον ViewMatrix για να εχει τις καινούργιες τιμές για το MVP.

ii) Περιστροφή γύρω από τον άξονα z με τα πλήκτρα Q και Z

Όμοια με την περιστροφή στον άξονα X γίνεται και η περιστροφή στον άξονα Z. Συγκεκριμένα το μόνο που αλλάζει είναι το vector που χρησιμοποιούμε στην rotate.Με το πλήκτρο Q κινείται δεξιόστροφα στον άξονα Z η κάμερα μας αν χρησιμοποιήσουμε το vector(0,0,1). Με το πλήκτρο Z κινείται αριστερόστροφα στον άξονα Z η κάμερα μας αν χρησιμοποιήσουμε το vector(0,0,-1).

iii) Zoom In -Zoom Out

```
float zoom = 45.0;//ftiaxame variable float zoom gia na kratisoume to arxiko radians tou ProjectionMatrix kai na to peirazoume analogws
```

Zoom in/zoom out με κατεύθυνση το σημείο P του πρίσματος με τα πλήκτρα<+> και <-> του numerical keypad του πληκτρολογίου. Σε αυτό το κομμάτι πειράζουμε το ProjectionMatrix και συγκεκριμένα το όρισμα της γωνίας. Η αρχική μας γωνιά είναι radians(45.0) και κάθε φορά που θελουμε να κάνουμε Zoom in αφαιρούμε 0.1 ενώ στο zoom out προσθέτουμε 0.1.

Τέλος χρειάστηκε να υλοποιήσουμε το MVP μέσα στο do :

```
//*********************************************
// Na prostethei kwdikas gia ton upologismo tou neo MVP
camera_function();//Kalw thn camera_function() h opoia allazei to ViewMatrix  & to projectionMatrix
View = getViewMatrix();//eisagw sthn view to allagmeno ViewMatrix an patithike kapoio apo ta apodekta plhktra
Projection = getProjectionMatrix(); //eisagw sthn Projection to allagmeno ProjectionMatrix an patithike kapoio apo ta apodekta plhktra


MVP = Projection * View * Model;// ypologizw to kainourgio mvp symfwna me ta kainourgia dedomena

// *********************************************/
glUniformMatrix4fv(MatrixID, 1, GL_FALSE, &MVP[0][0]);
```

Αρχικά καλούμε την camera_function() η οποία υλοποιεί την επιθυμητή λειτουργία της κάμερας. Στην συνέχεια ανανεώνουμε τα περιεχόμενα της View και του Projection σύμφωνα με τις τροποποιήσεις που έγιναν στην camera_function() και υπολογιζουμε το καινούργιο MVP.

2.Πληροφορίες σχετικά με την υλοποίηση

Η άσκηση υλοποιήθηκε στο λειτουργικό σύστημα Linux χρησιμοποιώντας το για την παραγωγή του κώδικα. Η εκτέλεση έγινε στο terminal με τις εντολές **make Source-1B** και ./**Source-1B** αξιοποιώντας το makefile του πρώτου εργαστηρίου. Επιλέχθηκε σε σχέση με το Visual studio διότι στα windows η srand δεν δημιουργούσε random τιμές μετά την πρώτη κλήση της ενώ στο Ubuntu έτρεχε εντελώς τυχαία σε κάθε κλήση της.

3.Σύντομη αξιολόγηση λειτουργίας ομάδας

Εργαστήκαμε πλήρως συνεργατικά στον σχεδιασμό και την υλοποίηση των ζητουμένων της άσκησης χωρίς να αντιμετωπίσουμε προβλήματα παρά μόνο ανταλλαγή σκέψεων για την ολοκλήρωση της άσκησης.

4.Αναφορές-Πηγές

https://glm.g-truc.net/0.9.4/api/a00206.html

Χρησιμοποιήθηκε για να μάθουμε πως ορίζεται η rotate και η scale.

https://learnopengl.com/Getting-started/Camera

Χρησιμοποιήθηκε για να μάθουμε τον τρόπο λειτουργίας της κάμερας.