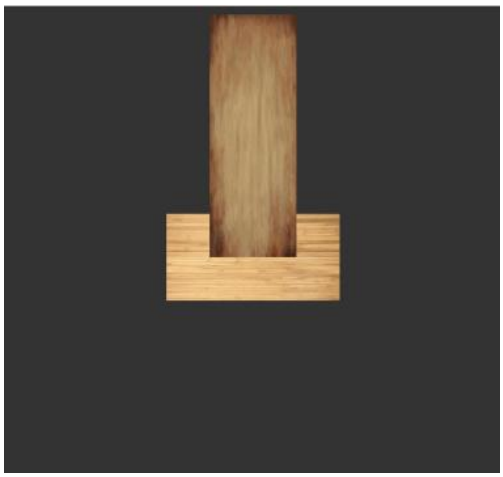
UNIVERSITY of IOANNINA

ΕΡΓΑΣΙΑ 1C

Σκοπός της άσκησης είναι η δημιουργία μιας εφαρμογής στην οποία θα υλοποιούνται Boolean λειτουργίες πάνω σε 3Δ στερεά.



i.

Στην γραμμή 208 του κώδικα με την εντολή αυτή καθορίζονται τα χαρακτηριστικά του παραθύρου ,όπως ζητείται και από την άσκηση με διαστάσεις 950x950 και με τίτλο <<Εργασία 1Γ – CSGBoolean Operations>>.

```
206
207     // Open a window and create its OpenGL context
208     window = glfwCreateWindow(950, 950, "Εργασία 1Γ – CSG-Boolean Operations", NULL, NULL);
209
```

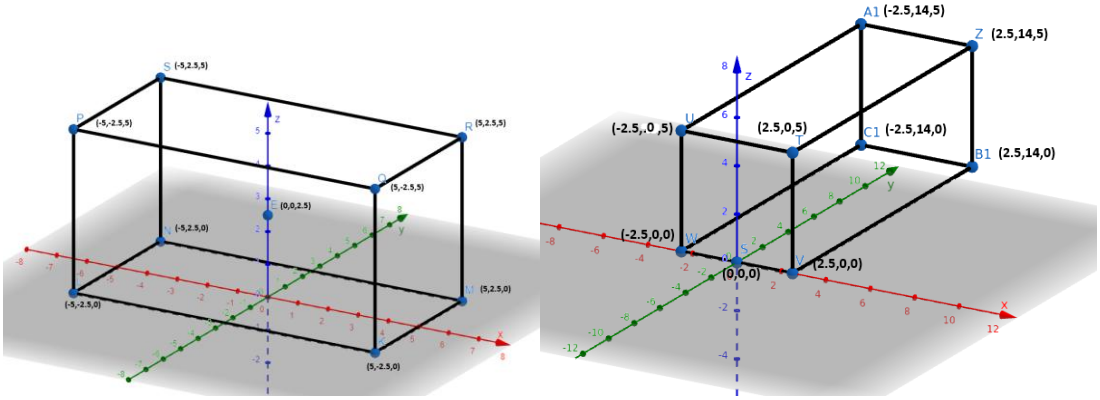
Η εντολή στην γραμμή 232 παίρνει σαν ορίσματα 4 νούμερα, εκ των οποίων τα τρία πρώτα που μας ενδιαφέρουν είναι τα rgb. Συνεπώς για την απόχρωση του σκούρου γκρι background του παραθύρου δίνουμε τις τιμές 0.3f, 0.3f, 0.3f, 1.0f .

```
//grey background
glClearColor(0.3f, 0.3f, 0.3f, 1.0f);
```

Με το πλήκτρο space η εφαρμογή τερματίζει.

```
1402     while (glfwGetKey(window, GLFW_KEY_SPACE) != GLFW_PRESS &&
1403            glfwWindowShouldClose(window) == 0);
```

ii . Τα αρχικά μας σχήματα Α και Β είναι ακριβώς τα ιδιά με την προηγουμένη άσκηση 1Β .



Το πρόγραμμα αρχίζει με τον σχεδιασμό δύο ορθογώνιων παραλληλεπιπέδων (Α και Β) και ενός κύβου. Το πρώτο ορθογώνιο παραλληλεπίπεδο, Α, τοποθετείται κατά μήκος του άξονα x και διαθέτει διαστάσεις με μήκος (k) ίσο με 10, ύψος (h) ίσο με 5, και πλάτος (w) ίσο με 5. Το κέντρο βάρους του Α εντοπίζεται στο σημείο Ε(0, 0, 2.5).

Το ορθογώνιο Α "τέμνεται" από το δεύτερο ορθογώνιο παραλληλεπίπεδο, Β, με διαστάσεις μήκους (m) ίσο με 5, ύψους (n) ίσο με 14, και πλάτος (v) ίσο με 5. Το Β τοποθετείται κατά μήκος του άξονα y, ενώ η μία πλευρά του είναι επάνω στο επίπεδο xz με σημείο Ο(0, 0, 0) ως κέντρο αυτής της πλευράς.

Για τον υπολογισμό των συντεταγμένων των σχημάτων Α και Β, εφαρμόσαμε μια μέθοδο που περιελάμβανε την αναπαράσταση των κορυφών τους σε ένα επίπεδο για ευκολότερη κατανόηση των σχημάτων και των συντεταγμένων. Επιλέξαμε το GeoGebra Online ως το κατάλληλο εργαλείο για την εισαγωγή και αναπαράσταση των συντεταγμένων των κορυφών, προκειμένου να επιβεβαιώσουμε την ορθότητα των διαστάσεων.

Αξιοποιήσαμε το Paint για να τοποθετήσουμε γραφικά τις συντεταγμένες των κορυφών, ενισχύοντας έτσι την οπτική αναπαράσταση των σχημάτων. Αυτή η διαδικασία λειτούργησε ως αξιόπιστη επιβεβαίωση των υπολογισμένων συντεταγμένων σε σχέση με τους άξονες X, Y, Z και τα κέντρα των σχημάτων.

Φτιάξαμε ένα buffer για το σχήμα A (figure):

```
static const GLfloat figure[] = {


    -5.0f, -2.5f, 0.0f,  //katw
     -5.0f, -2.5f, 5.0f,
     5.0f, -2.5f, 0.0f,
    5.0f, -2.5f, 5.0f,
    5.0f, -2.5f, 0.0f,
    -5.0f, -2.5f, 5.0f,
     -5.0f, -2.5f, 5.0f, //mprosta
     -5.0f, 2.5f, 5.0f,
     5.0f, -2.5f, 5.0f,
     5.0f, -2.5f, 5.0f,
     -5.0f, 2.5f, 5.0f,
     5.0f, 2.5f, 5.0f,
     -5.0f, 2.5f, 0.0f,  //plai aristera
     -5.0f, 2.5f, 5.0f,
     -5.0f, -2.5f, 5.0f,
    -5.0f, -2.5f, 5.0f,
     -5.0f, -2.5f, 0.0f,
     -5.0f, 2.5f, 0.0f,
     5.0f, -2.5f, 0.0f,  //plai dejia
     5.0f, -2.5f, 5.0f,
     5.0f, 2.5f, 0.0f,
    5.0f, -2.5f, 5.0f,
     5.0f, 2.5f, 0.0f,
     5.0f, 2.5f, 5.0f,
      -5.0f, 2.5f, 0.0f,
     5.0f, -2.5f, 0.0f,
     5.0f, 2.5f, 0.0f,
    -5.0f, -2.5f, 0.0f,  //  pisw
     -5.0f, 2.5f, 0.0f,
     5.0f, -2.5f, 0.0f,
    -5.0f, 2.5f, 0.0f,
     -5.0f, 2.5f, 5.0f,  //panw
     5.0f, 2.5f, 0.0f,
     -5.0f, 2.5f, 5.0f,
     5.0f, 2.5f, 0.0f,
     5.0f, 2.5f, 5.0f,

};
```

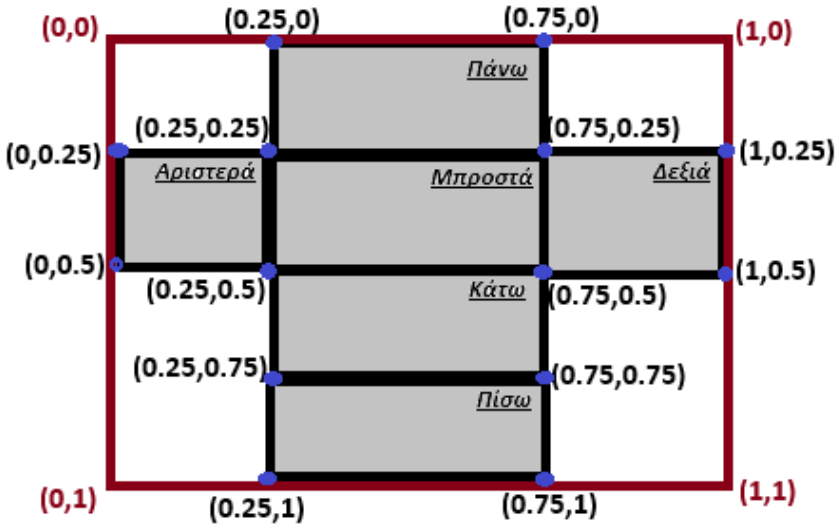
Φτιάξαμε ένα buffer για το σχήμα B (figure2):

```
static const GLfloat figure2[] = {

    -2.5f, 14.0f, 5.0f,//PANW
    -2.5f, 14.0f, 0.0f,
    2.5f, 14.0f, 5.0f,
    -2.5f, 14.0f, 0.0f,
    2.5f, 14.0f, 5.0f,
    2.5f, 14.0f, 0.0f,
    -2.5f, 0.0f, 5.0f,
    -2.5f, 0.0f, 0.0f,  //PLAINH aristera
    -2.5f, 14.0f, 5.0f,
    -2.5f, 14.0f, 0.0f,
    -2.5f, 0.0f, 0.0f,
    -2.5f, 14.0f, 5.0f,
    2.5f, 0.0f, 0.0f,
    2.5f, 14.0f, 0.0f,  //pisw
    -2.5f, 0.0f, 0.0f,
    -2.5f, 0.0f, 0.0f,
    -2.5f, 14.0f, 0.0f,
    2.5f, 14.0f, 0.0f,
    -2.5f, 0.0f, 5.0f, //mprosta
    -2.5f, 14.0f, 5.0f,
    2.5f, 0.0f, 5.0f,
    2.5f, 0.0f, 5.0f,
    2.5f, 14.0f, 5.0f,
    -2.5f, 14.0f, 5.0f,
    2.5f, 0.0f, 0.0f,
    2.5f, 0.0f, 5.0f, //PLAHNH dejia
    2.5f, 14.0f, 0.0f,
     2.5f, 0.0f, 5.0f,
    2.5f, 14.0f, 0.0f,
    2.5f, 14.0f, 5.0f,
    -2.5f, 0.0f, 0.0f,  //katv
    -2.5f, 0.0f, 5.0f,
    2.5f, 0.0f, 0.0f,
    2.5f, 0.0f, 0.0f,
    2.5f, 0.0f, 5.0f,
    -2.5f, 0.0f, 5.0f,
```

iii.

Στα μοντέλα Α και Β θα εφαρμόσαμε τις υφές που μας δόθηκαν τα αντίστοιχα αρχεία. Υπολογίσαμε τις uv συντεταγμένες των αντικειμένων κάθε φορά ξεχωριστά .

Unwrapping του σχήματος Α :

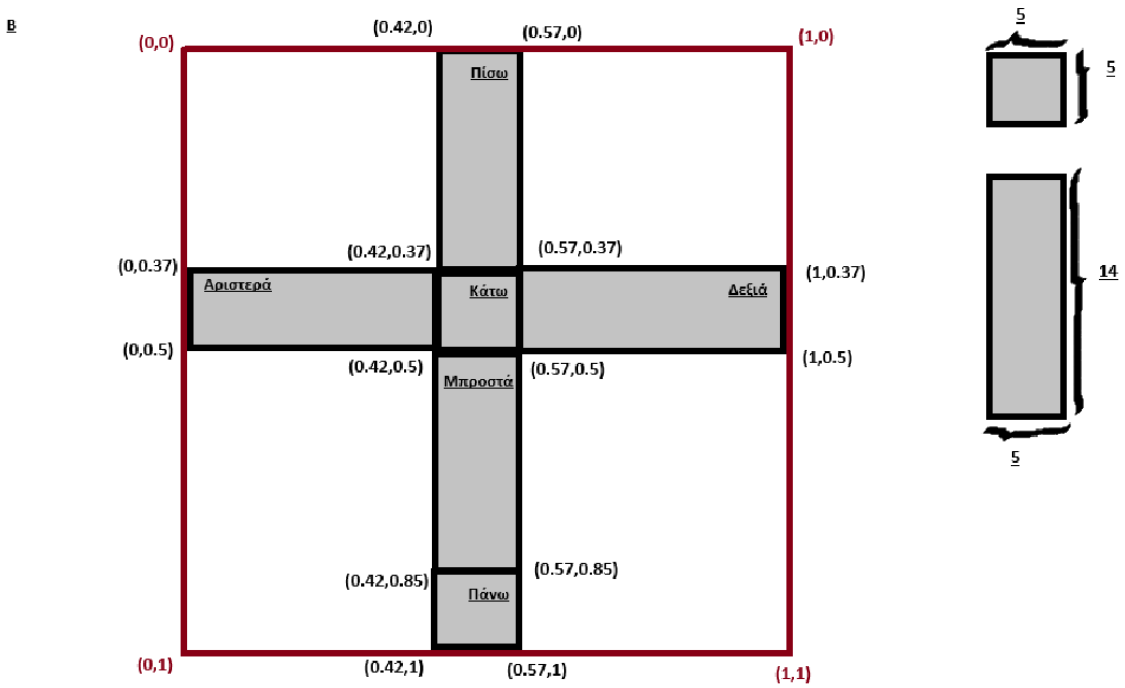


Πιο αναλυτικά οι πράξεις που κάναμε για να βρούμε τις UV συντεταγμένες είναι οι εξής :

Για τον άξονα V αθροίζουμε τις πλευρές των 4 παραλληλεπίπεδων (5+5+5+5=20) και διαιρούμε το 1 με το πλήθος των κορυφών (5) και άρα οι κορυφές απέχουν μεταξύ τους κατά 0,25 .

Για τον άξονα U αθροίζουμε τις πλευρές 2 τετράγωνων και ενός παραλληλεπίπεδου (5+5+10=20) . Δεδομένου ότι το μήκος του παραλληλεπιπέδου είναι 10, ο λόγος μεταξύ του και της πλευράς του τετραγώνου είναι ½.Οπότε διαιρούμε το 1 με το πλήθος των κορυφών (5) και άρα οι κορυφές απέχουν μεταξύ τους κατά 0,25 .

Unwrapping του σχήματος Β :



Πιο αναλυτικά οι πράξεις που κάναμε για να βρούμε τις UV συντεταγμένες είναι οι εξής :

Για τον άξονα V αθροίζουμε τις πλευρές 2 τετράγωνων και 2 παραλληλεπίπεδων (5+5+14+14=38).Διαιρούμε το 1 με το συνολικό άθροισμα(38)και άρα οι κορυφές απέχουν μεταξύ τους κατά 0,027 .

Για τον άξονα U αθροίζουμε τις πλευρές 2 παραλληλεπίπεδων και ενός τετραγωνου (5+14+14=33) . Διαιρούμε το 1 με το συνολικό άθροισμα(33) και άρα οι κορυφές απέχουν μεταξύ τους κατά 0,03 .

Ακολουθούν οι συντεταγμένες UV για το A που τοποθετήθηκαν μέσα στον κώδικα μας :

```
static const GLfloat uv_buffer1[] = {

    0.25f, 0.75f,
    0.25f, 0.5f,
    0.75f, 0.75f,
    0.75f, 0.5f,
    0.75f, 0.75f,
    0.25f, 0.5f,
    0.25f,0.5f, //mprosta
    0.25f,0.25f,
    0.75f,0.5f,
    0.75f,0.5f,
    0.25f,0.25f,
    0.75f,0.25f,
    0.0f, 0.25f,  // plai aristera
    0.25f, 0.25f,
    0.25f, 0.5f,
    0.25f, 0.5f,
    0.0f, 0.5f,
    0.0f, 0.25f,
    1.0f, 0.5f, //plai dejia
    0.75f, 0.5f,
    1.0f, 0.25f,
    0.75f, 0.5f,
    1.0f, 0.25f,
    0.75f, 0.25f,
    0.25f, 1.0f, //katw
    0.75f, 0.75f,
    0.75f, 1.0f,
    0.25, 0.75f,
    0.25f, 1.0f,
    0.75f, 0.75f,
    0.25f, 0.0f, //panw
    0.25f, 0.25f,
    0.75f, 0.0f,
    0.25f, 0.25f,
    0.75f, 0.0f,
    0.75f, 0.25f,

```

Ακολουθούν οι συντεταγμένες UV για το B που τοποθετήθηκαν μέσα στον κώδικα μας :

```
static const GLfloat uv_buffer2[] = {

    0.57f, 0.42f,
    0.57f, 0.57f,
    0.42f, 0.42f,
    0.57f, 0.57f,
    0.42f, 0.42f,
    0.42f, 0.57f,
    0.0f, 0.5f, //aristera
    0.42f, 0.5f,
    0.0f, 0.37f,
    0.42f, 0.37f,
    0.42f, 0.5f,
    0.0f, 0.37f,
    0.42f, 0.37f,
    0.42f, 0.0f,
    0.57f, 0.37f,
    0.57f, 0.37f,
    0.57f, 0.0f,
    0.42f, 0.0f,
    0.42f, 0.57f,
    0.42f, 0.85f,
    0.57f, 0.57f,
    0.57f, 0.57f,
    0.57f, 0.85f,
    0.42f, 0.85f,
    1.0f, 0.5f,  //dejia
    1.0f, 0.37f,
    0.57f, 0.37f,
    0.57f, 0.5f,
    0.57f, 0.37f,
    1.0f, 0.5f,
    0.42, 0.5f,  //katw
    0.42f, 0.37f,
    0.57f, 0.5f,
    0.57f, 0.5f,
    0.57f, 0.37f,
    0.42, 0.5f,

```

Έξω από τον βρόχο DO-WHILE έχουμε κάνει load τα δυο texture . Πιο αναλυτικά:

```
13      int height,width,numChannels;
14      unsigned char* data = stbi_load("texture-model-A.jpg", &width, &height, &numChannels, 0);
15
16      int height1,width1,numChannels1;
17      unsigned char* data1 = stbi_load("texture-model-B.jpg", &width1, &height1, &numChannels1, 0);
18
19      GLuint textureID;
20      glGenTextures(1, &textureID);
21
```

Μέσα στον βρόχο DO-WHILE :

```
948     glBindTexture(GL_TEXTURE_2D, textureID);
949     glTexImage2D(GL_TEXTURE_2D, 0,GL_RGB, width, height, 0, GL_RGB, GL_UNSIGNED_BYTE, data);
950
951     glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST);
952     glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);
953
954
955     GLuint vertexbuffer;
956     glGenBuffers(1, &vertexbuffer);
957     glBindBuffer(GL_ARRAY_BUFFER, vertexbuffer);
958     glBufferData(GL_ARRAY_BUFFER, sizeof(figure), figure, GL_STATIC_DRAW);
959
960
961     GLuint uvbuffer;
962     glGenBuffers(1, &uvbuffer);
963     glBindBuffer(GL_ARRAY_BUFFER,uvbuffer);
964     glBufferData(GL_ARRAY_BUFFER, sizeof(uv_buffer1), uv_buffer1, GL_STATIC_DRAW);
965     // Clear the screen
966     glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
967
968     // Use our shader
969     glUseProgram(programID);
970
971
972     // 1rst attribute buffer : vertices
973     glEnableVertexAttribArray(0);
974     glBindBuffer(GL_ARRAY_BUFFER, vertexbuffer);
975     glVertexAttribPointer(
976         0,                  // attribute 0. No particular reason for 0, but must match the layout in the shader.
977         3,                  // size
978         GL_FLOAT,           // type
979         GL_FALSE,           // normalized?
980         0,                  // stride
981         (void*)0            // array buffer offset
982     );
983
984     glEnableVertexAttribArray(1);
985     glBindBuffer(GL_ARRAY_BUFFER, uvbuffer);
986     glVertexAttribPointer(
987         1,                  // attribute 0. No particular reason for 0, but must match the layout in the shader.
988         2,                  // size
989         GL_FLOAT,           // type
990         GL_FALSE,           // normalized?
991         0,                  // stride
992         (void*)0            // array buffer offset
993     );
994
995     glDrawArrays(GL_TRIANGLES, 0,36);
```

Χρησιμοποιεί τους buffers για τις κορυφές και τις συντεταγμένες του texture, ενεργοποιεί το shader και σχεδιάζει το σχήμα Α στην οθόνη.

```
        glBindTexture(GL_TEXTURE_2D, textureID);
        glTexImage2D(GL_TEXTURE_2D, 0,GL_RGB, width1, height1, 0, GL_RGB, GL_UNSIGNED_BYTE, data1);

        glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST);
        glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);


        GLuint vertexbuffer2;
        glGenBuffers(1, &vertexbuffer2);
        glBindBuffer(GL_ARRAY_BUFFER, vertexbuffer2);
        glBufferData(GL_ARRAY_BUFFER, sizeof(figure2), figure2, GL_STATIC_DRAW);

        GLuint uvbuffer3;
        glGenBuffers(1, &uvbuffer3);
        glBindBuffer(GL_ARRAY_BUFFER,uvbuffer3);
        glBufferData(GL_ARRAY_BUFFER, sizeof(uv_buffer2), uv_buffer2, GL_STATIC_DRAW);

        glEnableVertexAttribArray(0);
        glBindBuffer(GL_ARRAY_BUFFER, vertexbuffer2);
        glVertexAttribPointer(
            0,                  // attribute 0. No particular reason for 0, but must match the layout in the shader.
            3,                  // size
            GL_FLOAT,           // type
            GL_FALSE,           // normalized?
            0,                  // stride
            (void*)0            // array buffer offset
        );

        glEnableVertexAttribArray(1);
        glBindBuffer(GL_ARRAY_BUFFER, uvbuffer3);
        glVertexAttribPointer(
            1,                  // attribute 0. No particular reason for 0, but must match the layout in the shader.
            2,                  // size
            GL_FLOAT,           // type
            GL_FALSE,           // normalized?
            0,                  // stride
            (void*)0            // array buffer offset
        );



        glDrawArrays(GL_TRIANGLES, 0,36);
```

Χρησιμοποιεί τους buffers για τις κορυφές και τις συντεταγμένες του texture, ενεργοποιεί το shader και σχεδιάζει το σχήμα B στην οθόνη.

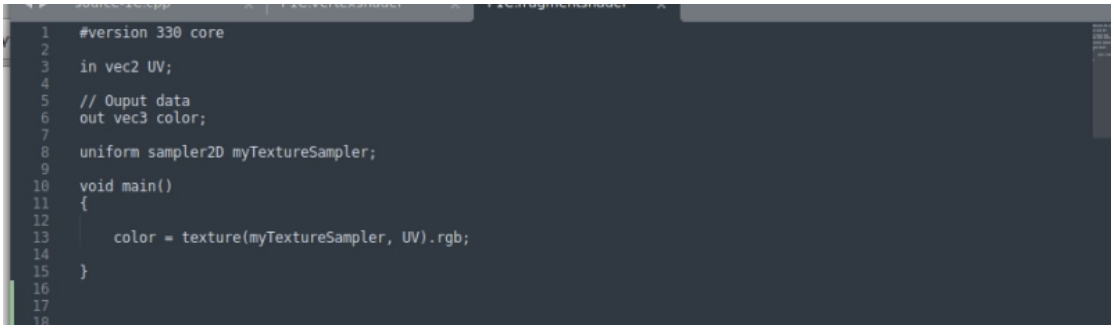
Αρχεία

```
#version 330 core

// Input vertex data, different for all executions of this shader.
layout(location = 0) in vec3 vertexPosition_modelspace;
layout(location = 1) in vec2 vertexUV;

// Output data ; will be interpolated for each fragment.
out vec2 UV;

uniform mat4 MVP;

void main(){

        gl_Position =  MVP * vec4(vertexPosition_modelspace,1);
        UV = vertexUV;
}
```

In vec3: αναπαριστά τις τρισδιάστατες συντεταγμένες των κορυφών του μοντέλου

In vec2: αναπαριστά τις δισδιάστατες συντεταγμένες UV

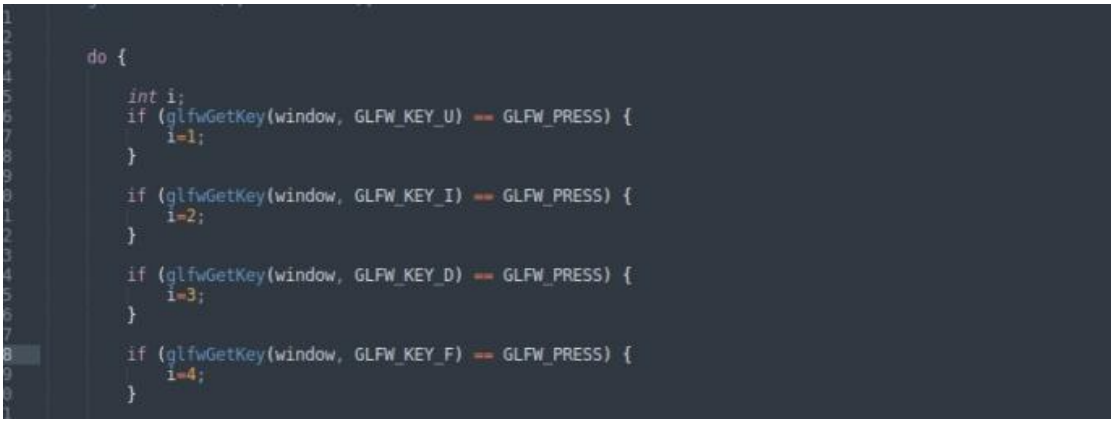
Η μεταβλητή UV είναι μια out μεταβλητή που αντιστοιχεί στις συντεταγμένες UV του texture που θα χρησιμοποιηθούν ως είσοδοι στο αρχείο fragmentshader.



```
#version 330 core

in vec2 UV;

// Ouput data
out vec3 color;

uniform sampler2D myTextureSampler;

void main()
{

    color = texture(myTextureSampler, UV).rgb;

}
```

iv.

<u>Μέσα στον βρόχο DO-WHILE :</u>



```
do {

    int i;
    if (glfwGetKey(window, GLFW_KEY_U) == GLFW_PRESS) {
        i=1;
    }

    if (glfwGetKey(window, GLFW_KEY_I) == GLFW_PRESS) {
        i=2;
    }

    if (glfwGetKey(window, GLFW_KEY_D) == GLFW_PRESS) {
        i=3;
    }

    if (glfwGetKey(window, GLFW_KEY_F) == GLFW_PRESS) {
        i=4;
    }
```

$\boldsymbol{A} \cup \boldsymbol{B}$ - Ένωση των A και B (Union) με το πλήκτρο <u>.

• $\boldsymbol{A} \cap \boldsymbol{B}$ - Τομή των A και B (Intersection) με το πλήκτρο <i>.

• $\boldsymbol{A} - \boldsymbol{B}$ Διαφορά (Difference) με το πλήκτρο <d>.

• $\boldsymbol{B} - \boldsymbol{A}$ Διαφορά (Difference) με το πλήκτρο <f>.

Για κάθε κουμπί ορίζεται ένα **flag i** με διαφορετική τιμή κάθε φορά με αποτέλεσμα πατώντας το αντίστοιχο κουμπί να οδηγείται στην κατάσταση της κάθε λειτουργίας .

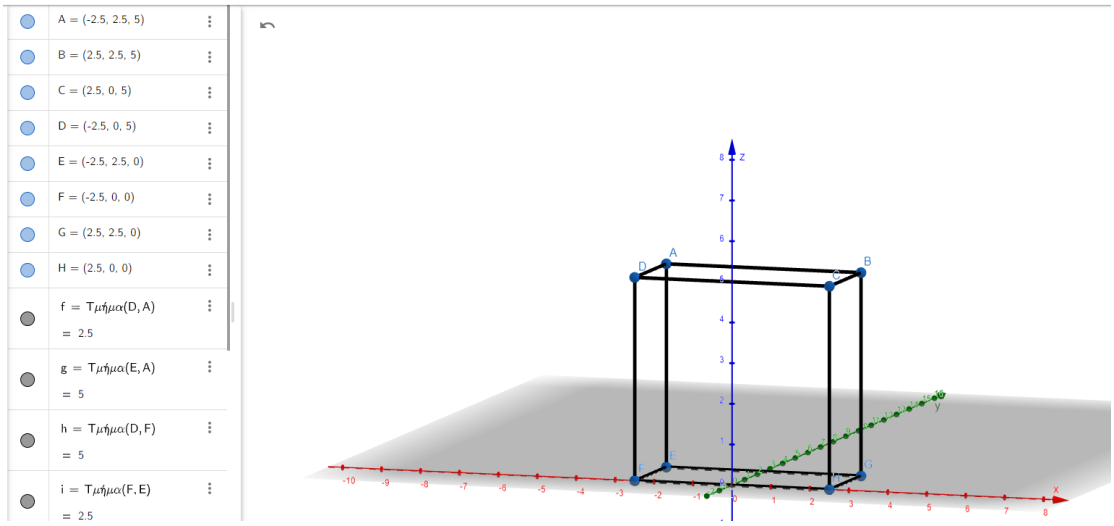
<u>$\boldsymbol{A} \cup \boldsymbol{B}$ - Ένωση των A και B (Union)</u>

Για την ένωση του A και B χρησιμοποιούμε τους ίδιους buffers του ερωτήματος 3 με την διαφορά ότι εφαρμόζουμε και στα δυο σχήματα την εικόνα A.

$\boldsymbol{A \cap B}$ - Τομή των Α και Β (Intersection)

Εκτελέσαμε τις πράξεις για να βρούμε τις συντεταγμένες του σχήματος $\boldsymbol{A \cap B}$ ως εξής :

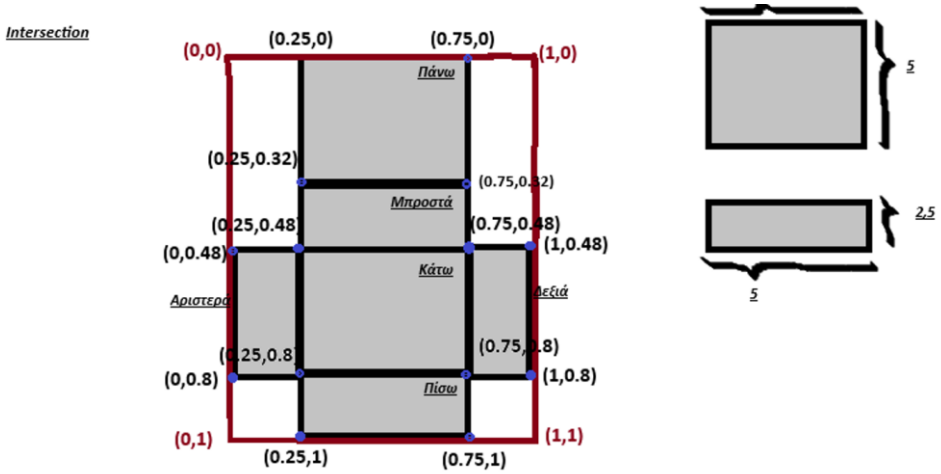
Συγκρίνουμε κάθε στοιχείο του συνόλου Α με κάθε στοιχείο του συνόλου Β για να δούμε ποια στοιχεία είναι κοινά σε αυτά τα δύο σύνολα.



Φτιάξαμε ένα buffer για το σχήμα (inter):

```
static const GLfloat inter[] = {

    -2.5f, 2.5f, 5.0f,
    2.5f, 2.5f, 5.0f,
    2.5f, 0.0f, 5.0f,
    2.5f, 0.0f, 5.0f,
    -2.5f, 0.0f, 5.0f,
    -2.5f, 2.5f, 5.0f,
    -2.5f, 2.5f, 0.0f,
    -2.5f, 2.5f, 5.0f,
    -2.5f, 0.0f, 5.0f,
    -2.5f, 0.0f, 5.0f,
    -2.5f, 0.0f, 0.0f,
    -2.5f, 2.5f, 0.0f,
    2.5f, 2.5f, 0.0f,
    -2.5f, 2.5f, 0.0f,
    -2.5f, 0.0f, 0.0f,
    -2.5f, 0.0f, 0.0f,
    2.5f, 0.0f, 0.0f,
    2.5f, 2.5f, 0.0f,
    2.5f, 2.5f, 5.0f,
    2.5f, 2.5f, 0.0f,
    2.5f, 0.0f, 0.0f,
    2.5f, 0.0f, 0.0f,
    2.5f, 0.0f, 5.0f,
    2.5f, 2.5f, 5.0f,
    -2.5f, 0.0f, 5.0f,
    -2.5f, 0.0f, 0.0f,
    2.5f, 0.0f, 0.0f,
    2.5f, 0.0f, 0.0f,
    2.5f, 0.0f, 5.0f,
    -2.5f, 0.0f, 5.0f,
    2.5f, 2.5f, 5.0f,
    -2.5f, 2.5f, 5.0f
    -2.5f, 2.5f, 0.0f,
    -2.5f, 2.5f, 0.0f,
    2.5f, 2.5f, 0.0f,
    2.5f, 2.5f, 5.0f
```

Unwrapping του σχήματος A ∩ B :



Για τον άξονα V αθροίζουμε τις πλευρές 2 τετράγωνων και 2 παραλληλεπίπεδων (5+5+2.5+2.5=15). Δεδομένου ότι το μήκος του τετράγωνου είναι 5, ο λόγος μεταξύ του και της πλευράς του παραλληλεπίπεδου είναι ½.Δηλαδη μετατρέψαμε το τετράγωνο σε 2 παραλληλεπίπεδα για να υπολογίσουμε το πλήθος των κορυφών και αντίστοιχα έγινε και στο δεύτερο τετράγωνο η ιδιά διαδικασία. Οπότε διαιρούμε το 1 με το πλήθος των κορυφών (7) και άρα οι κορυφές απέχουν μεταξύ τους κατά 0,14285714. Στρογγυλοποιήσαμε στο 0,16 έτσι ώστε αθροίζοντας τις 7 κορυφές να φτάσουμε όσο γίνεται πιο κοντά στο 1 .

Για τον άξονα U αθροίζουμε τις πλευρές 2 παραλληλεπίπεδων και ενός τετράγωνου (2,5+2,5+5=10) . Δεδομένου ότι το μήκος του τετράγωνου είναι 5, ο λόγος μεταξύ του και της πλευράς του παραλληλεπίπεδου είναι ½.Δηλαδη μετατρέψαμε το τετράγωνο σε 2 παραλληλεπίπεδα για να υπολογίσουμε το πλήθος των κορυφών και αντίστοιχα έγινε και στο δεύτερο τετράγωνο. Οπότε διαιρούμε το 1 με το πλήθος των κορυφών (5) και άρα οι κορυφές απέχουν μεταξύ τους κατά 0,25.

Ακολουθούν οι συντεταγμένες UV που τοποθετήθηκαν μέσα στον κώδικα μας :

```
static const GLfloat uv_buffer9[] = {

0.25f, 0.32f,
0.75f, 0.32f,
0.75f, 0.48f,
0.75f, 0.48f,
0.25f,0.48f,
0.25f, 0.32f,
0.0f, 0.48f,  //plai aristera
0.25f, 0.48f,
0.25f, 0.8f,
0.25f, 0.8f,
0.0f, 0.8f,
0.0f, 0.48f,
0.25f, 0.8f,  //pisw
0.75f, 0.8f,
0.75f, 1.0f,
0.75f, 1.0f,
0.25f, 1.0f,
0.25f, 0.8f,
0.75f, 0.48f,  //plai dejia
1.0f, 0.48f,
1.0f, 0.8f,
1.0f, 0.8f,
0.75f, 0.8f,
0.75f, 0.48f,
0.75f, 0.8f,  //katw
0.75f, 0.48f,
0.25f, 0.48f,
0.25f, 0.48f,
0.25f, 0.8f,
0.75f, 0.8f,
0.75f, 0.0f,
0.25f, 0.0f,
0.25f, 0.32f,
0.25f, 0.32f,
0.75f, 0.32f,
0.75f, 0.0f,
```

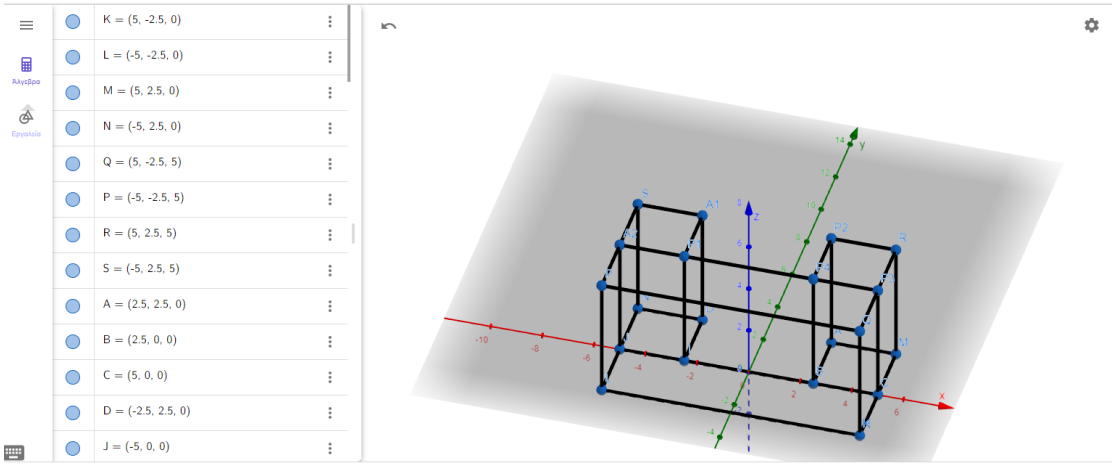
εφαρμόζουμε στο σχήμα την εικόνα Α.

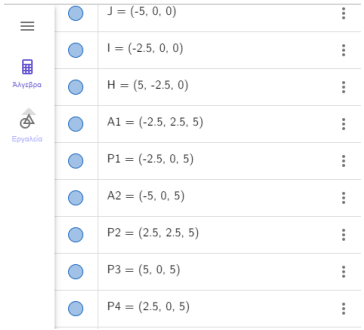
```
if (i==2) {
        glBindTexture(GL_TEXTURE_2D, textureID);
        glTexImage2D(GL_TEXTURE_2D, 0,GL_RGB, width, height, 0, GL_RGB, GL_UNSIGNED_BYTE, data);

        glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST);
        glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);


        GLuint vertexbuffer10;
        glGenBuffers(1, &vertexbuffer10);
        glBindBuffer(GL_ARRAY_BUFFER, vertexbuffer10);
        glBufferData(GL_ARRAY_BUFFER, sizeof(inter), inter, GL_STATIC_DRAW);


        GLuint uvbuffer8;
        glGenBuffers(1, &uvbuffer8);
        glBindBuffer(GL_ARRAY_BUFFER,uvbuffer8);
        glBufferData(GL_ARRAY_BUFFER, sizeof(uv_buffer9), uv_buffer9, GL_STATIC_DRAW);
        // Clear the screen
        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

        // Use our shader
        glUseProgram(programID);


        // 1rst attribute buffer : vertices
        glEnableVertexAttribArray(0);
        glBindBuffer(GL_ARRAY_BUFFER, vertexbuffer10);
        glVertexAttribPointer(
            0,                  // attribute 0. No particular reason for 0, but must match the layout in the shader.
            3,                  // size
            GL_FLOAT,           // type
            GL_FALSE,           // normalized?
            0,                  // stride
            (void*)0            // array buffer offset
        );

        glEnableVertexAttribArray(1);
        glBindBuffer(GL_ARRAY_BUFFER, uvbuffer8);
        glVertexAttribPointer(
            1,                  // attribute 0. No particular reason for 0, but must match the layout in the shader.
            2,                  // size
            GL_FLOAT,           // type
            GL_FALSE,           // normalized?
            0,                  // stride
            (void*)0            // array buffer offset
        );
        glDrawArrays(GL_TRIANGLES, 0,36);
```

A **– *B* Διαφορά (Difference)**

Εκτελέσαμε τις πράξεις για να βρούμε τις συντεταγμένες του σχήματος B-A ως εξής :

(A.x – B.x , A.y – B.y , A.z – B.z)

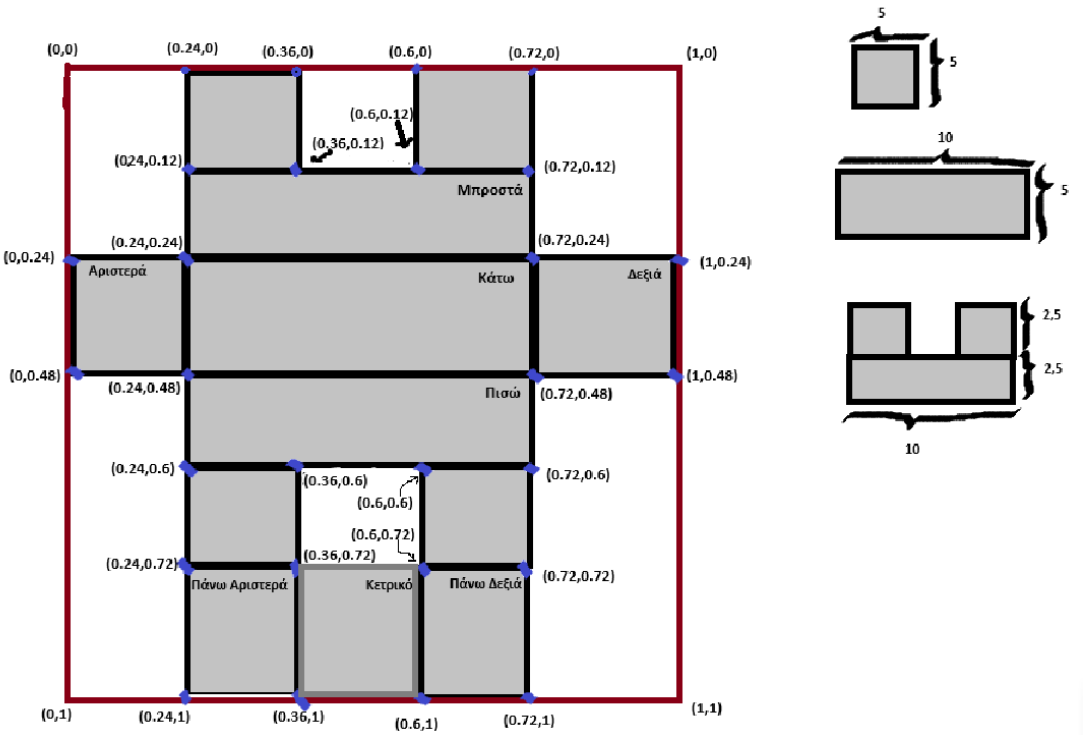




<u>Φτιάξαμε ένα buffer με τις συντεταγμένες για το σχήμα :</u>

```
static const GLfloat DIFF1[] = {
    -5.0f, -2.5f, 0.0f,  //katw
     -5.0f, -2.5f, 5.0f,
     5.0f, -2.5f, 0.0f,
     5.0f, -2.5f, 5.0f,
     5.0f, -2.5f, 0.0f,
     -5.0f, -2.5f, 5.0f,
     -5.0f, 2.5f, 0.0f,  //plai aristera
     -5.0f, 2.5f, 5.0f,
     -5.0f, -2.5f, 5.0f,
     -5.0f, -2.5f, 5.0f,
     -5.0f, -2.5f, 0.0f,
     -5.0f, 2.5f, 0.0f,
     5.0f, -2.5f, 0.0f,  //plai dejia
     5.0f, -2.5f, 5.0f,
     5.0f, 2.5f, 0.0f,
     5.0f, -2.5f, 5.0f,
     5.0f, 2.5f, 0.0f,
     5.0f, 2.5f, 5.0f,
     5.0f, 2.5f, 0.0f, //pisw tetr aristera
     2.5f, 2.5f, 0.0f,
     2.5f, 0.0f, 0.0f,
     2.5f, 0.0f, 0.0f,
     5.0f, 0.0f, 0.0f,
     5.0f, 2.5f, 0.0f,
     -2.5f, 2.5f, 0.0f, //tetr dejia
     -5.0f, 2.5f, 0.0f,
     -5.0f, 0.0f, 0.0f,
     -5.0f, 0.0f, 0.0f,
     -2.5f, 0.0f, 0.0f,
     -2.5f, 2.5f, 0.0f,
     5.0f, 0.0f, 0.0f, //parallil
    -5.0f, 0.0f, 0.0f,
    -5.0f, -2.5f, 0.0f,
    -5.0f, -2.5f, 0.0f,
     5.0f, -2.5f, 0.0f,
     5.0f, 0.0f, 0.0f,
     -5.0f, 2.5f, 5.0f, //mprosta
     -2.5f, 2.5f, 5.0f,
     -2.5f, 0.0f, 5.0f,
     -2.5f, 0.0f, 5.0f,
     -5.0f, 0.0f, 5.0f,
     -5.0f, 2.5f, 5.0f,
     2.5f, 2.5f, 5.0f,
     5.0f, 2.5f, 5.0f,
     5.0f, 0.0f, 5.0f,
     -2.5f, 2.5f, 5.0f,
     -2.5f, 0.0f, 5.0f,
     -2.5f, 0.0f, 5.0f,
     -5.0f, 0.0f, 5.0f,
     -5.0f, 2.5f, 5.0f,
     2.5f, 2.5f, 5.0f,
     5.0f, 2.5f, 5.0f,
     5.0f, 0.0f, 5.0f,
     5.0f, 0.0f, 5.0f,
     2.5f, 0.0f, 5.0f,
     2.5f, 2.5f, 5.0f,
     -5.0f, 0.0f, 5.0f,
      5.0f, 0.0f, 5.0f,
      5.0f, -2.5f, 5.0f,
      5.0f, -2.5f, 5.0f,
     -5.0f, -2.5f, 5.0f,
     -5.0f, 0.0f, 5.0f,
     -5.0f, 2.5f, 0.0f,//pano lo aristero paral
    -2.5f, 2.5f, 0.0f,
    -2.5f, 2.5f, 5.0f,
    -2.5f, 2.5f, 5.0f,
    -5.0f, 2.5f, 5.0f,
    -5.0f, 2.5f, 0.0f,
     2.5f, 2.5f, 0.0f,//pano deji paral
     5.0f, 2.5f, 0.0f,
     5.0f, 2.5f, 5.0f,
     5.0f, 2.5f, 5.0f,
     2.5f, 2.5f, 5.0f,
     2.5f, 2.5f, 0.0f,
     -2.5f, 0.0f, 0.0f,  //pano mesa kentriko te
     2.5f, 0.0f, 0.0f,
     2.5f, 0.0f, 5.0f,
     2.5f, 0.0f, 5.0f,
     -2.5f, 0.0f, 5.0f,
     -2.5f, 0.0f, 0.0f,
    -2.5f, 2.5f, 0.0f, //pano mesa aristero para
    -2.5f, 0.0f, 0.0f,
    -2.5f, 0.0f, 5.0f,
     -2.5f, 0.0f, 5.0f,
     -2.5f, 2.5f, 5.0f,
     -2.5f, 2.5f, 0.0f,
     2.5f, 0.0f, 0.0f,//dejio plaino mesa
     2.5f, 2.5f, 0.0f,
     2.5f, 2.5f, 5.0f,
     2.5f, 2.5f, 5.0f,
     2.5f, 0.0f, 5.0f,
     2.5f, 0.0f, 0.0f,
```

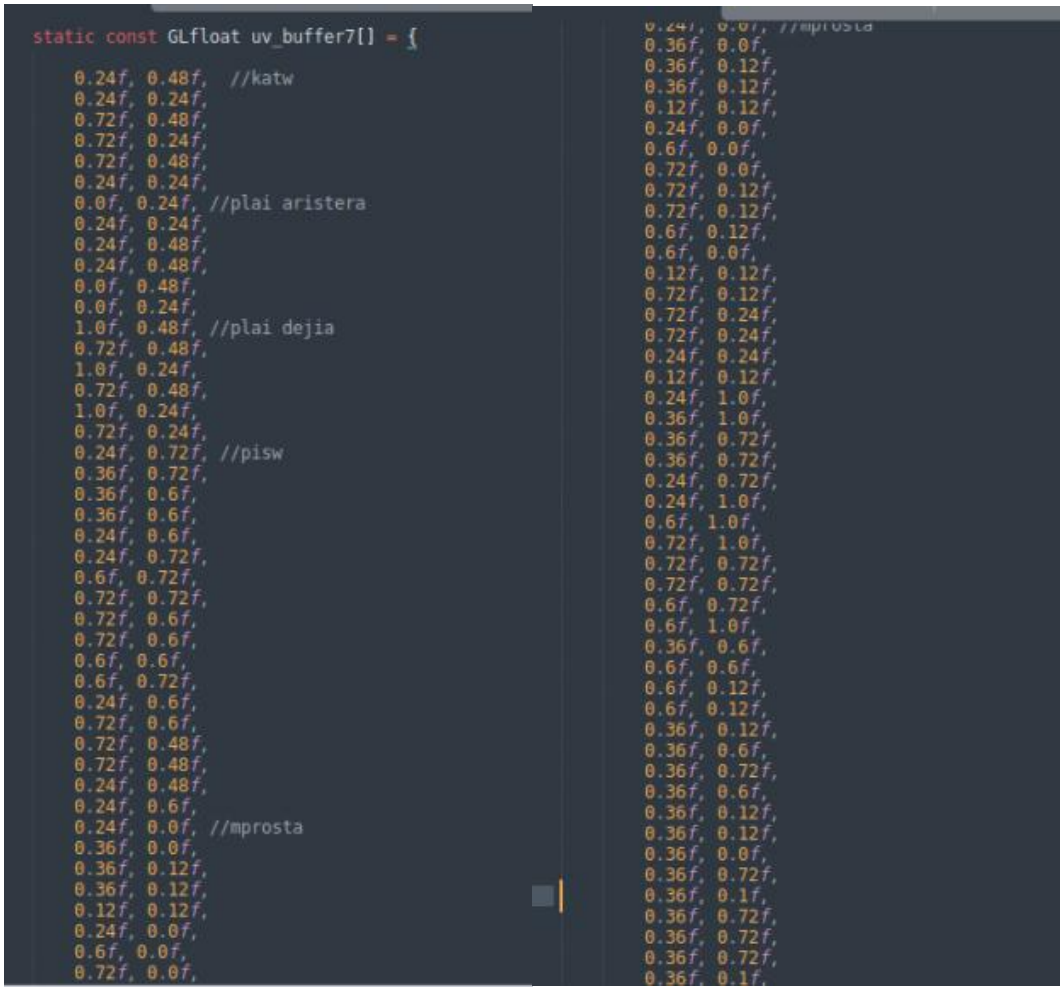
Unwrapping του σχήματος $A - B$:



Για τον άξονα U αθροίζουμε τις πλευρές 2 τετράγωνων και 1 παραλληλεπίπεδο (5+5+10=20). Δεδομένου ότι το μήκος του τετράγωνου είναι 5, ο λόγος μεταξύ του και της πλευράς του παραλληλεπίπεδου είναι ½.Δηλαδη μετατρέψαμε το παραλληλεπίπεδο σε 4 τετράγωνα για να υπολογίσουμε το πλήθος των κορυφών. Οπότε διαιρούμε το 1 με το πλήθος των κορυφών (9) και άρα οι κορυφές απέχουν μεταξύ τους κατά 0,12.

Για τον άξονα V αθροίζουμε τις πλευρές 2 παραλληλεπίπεδων και 2 τετράγωνων (5+5+5+5=20) και διαιρούμε το 1 με το πλήθος των κορυφών (5) και άρα οι κορυφές απέχουν μεταξύ τους κατά 0,25 .

Ακολουθούν οι συντεταγμένες UV που τοποθετήθηκαν μέσα στον κώδικα μας :



```
static const GLfloat uv_buffer7[] = {

    0.24f, 0.48f,  //katw
    0.24f, 0.24f,
    0.72f, 0.48f,
    0.72f, 0.24f,
    0.72f, 0.48f,
    0.24f, 0.24f,
    0.0f, 0.24f, //plai aristera
    0.24f, 0.24f,
    0.24f, 0.48f,
    0.24f, 0.48f,
    0.0f, 0.48f,
    0.0f, 0.24f,
    1.0f, 0.48f, //plai dejia
    0.72f, 0.48f,
    1.0f, 0.24f,
    0.72f, 0.48f,
    1.0f, 0.24f,
    0.72f, 0.24f,
    0.24f, 0.72f, //pisw
    0.36f, 0.72f,
    0.36f, 0.6f,
    0.36f, 0.6f,
    0.24f, 0.6f,
    0.24f, 0.72f,
    0.6f, 0.72f,
    0.72f, 0.72f,
    0.72f, 0.6f,
    0.72f, 0.6f,
    0.6f, 0.6f,
    0.6f, 0.72f,
    0.24f, 0.6f,
    0.72f, 0.6f,
    0.72f, 0.48f,
    0.72f, 0.48f,
    0.24f, 0.48f,
    0.24f, 0.6f,
    0.24f, 0.0f, //mprosta
    0.36f, 0.0f,
    0.36f, 0.12f,
    0.36f, 0.12f,
    0.12f, 0.12f,
    0.24f, 0.0f,
    0.6f, 0.0f,
    0.72f, 0.0f,
    0.36f, 0.0f,
    0.36f, 0.12f,
    0.36f, 0.12f,
    0.12f, 0.12f,
    0.24f, 0.0f,
    0.6f, 0.0f,
    0.72f, 0.0f,
    0.72f, 0.12f,
    0.72f, 0.12f,
    0.6f, 0.12f,
    0.6f, 0.0f,
    0.12f, 0.12f,
    0.72f, 0.12f,
    0.72f, 0.24f,
    0.72f, 0.24f,
    0.24f, 0.24f,
    0.12f, 0.12f,
    0.24f, 1.0f,
    0.36f, 1.0f,
    0.36f, 0.72f,
    0.36f, 0.72f,
    0.24f, 0.72f,
    0.24f, 1.0f,
    0.6f, 1.0f,
    0.72f, 1.0f,
    0.72f, 0.72f,
    0.72f, 0.72f,
    0.6f, 0.72f,
    0.6f, 1.0f,
    0.36f, 0.6f,
    0.6f, 0.6f,
    0.6f, 0.12f,
    0.6f, 0.12f,
    0.36f, 0.12f,
    0.36f, 0.6f,
    0.36f, 0.72f,
    0.36f, 0.6f,
    0.36f, 0.12f,
    0.36f, 0.12f,
    0.36f, 0.0f,
    0.36f, 0.72f,
    0.36f, 0.1f,
    0.36f, 0.72f,
    0.36f, 0.72f,
    0.36f, 0.72f,
    0.36f, 0.1f,
```

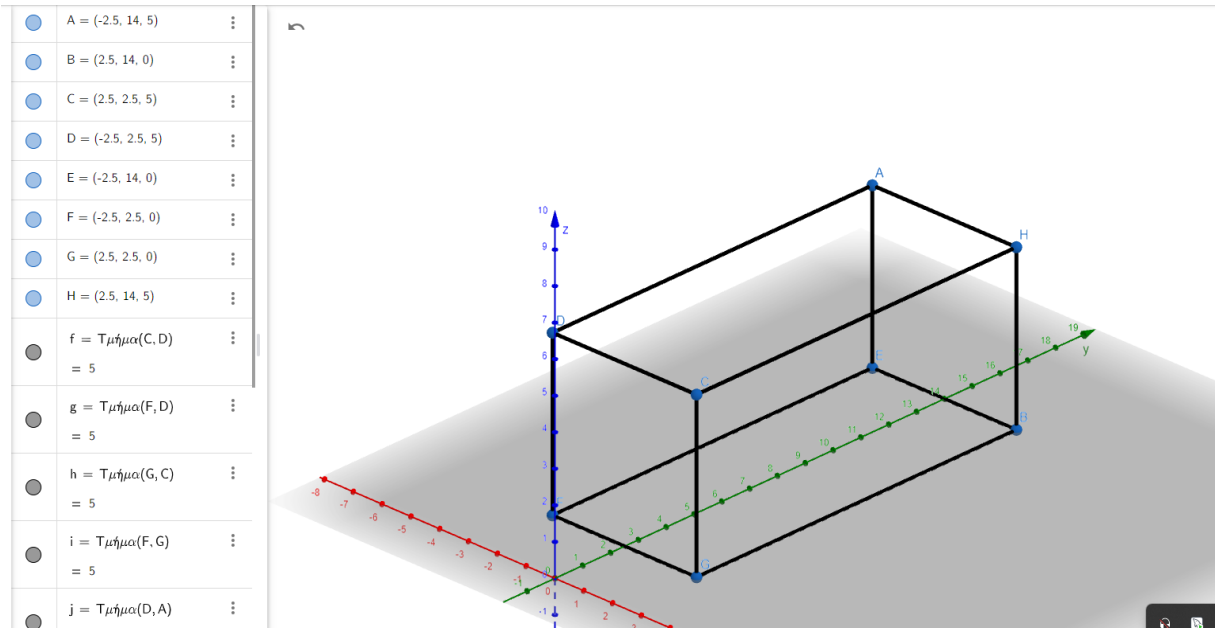
εφαρμόζουμε στο σχήμα την εικόνα Α.

```
if (i==3) {


    glBindTexture(GL_TEXTURE_2D, textureID);
    glTexImage2D(GL_TEXTURE_2D, 0,GL_RGB, width, height, 0, GL_RGB, GL_UNSIGNED_BYTE, data);

    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);

    GLuint vertexbuffer5;
    glGenBuffers(1, &vertexbuffer5);
    glBindBuffer(GL_ARRAY_BUFFER, vertexbuffer5);
    glBufferData(GL_ARRAY_BUFFER, sizeof(DIFF1), DIFF1, GL_STATIC_DRAW);

    GLuint uvbuffer6;
    glGenBuffers(1, &uvbuffer6);
    glBindBuffer(GL_ARRAY_BUFFER,uvbuffer6);
    glBufferData(GL_ARRAY_BUFFER, sizeof(uv_buffer7), uv_buffer7, GL_STATIC_DRAW);
    // Clear the screen
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

    // Use our shader
    glUseProgram(programID);


    // 1rst attribute buffer : vertices
    glEnableVertexAttribArray(0);
    glBindBuffer(GL_ARRAY_BUFFER, vertexbuffer5);
    glVertexAttribPointer(
        0,                  // attribute 0. No particular reason for 0, but must match the layout in the
        3,                  // size
        GL_FLOAT,           // type
        GL_FALSE,           // normalized?
        0,                  // stride
        (void*)0            // array buffer offset
        );

    glEnableVertexAttribArray(1);
    glBindBuffer(GL_ARRAY_BUFFER, uvbuffer6);
    glVertexAttribPointer(
        1,                  // attribute 0. No particular reason for 0, but must match the layout in the
        2,                  // size
        GL_FLOAT,           // type
        GL_FALSE,           // normalized?
        0,                  // stride
        (void*)0            // array buffer offset
        );

    glDrawArrays(GL_TRIANGLES, 0,132);
```

B – A **Διαφορά (Difference)**

Εκτελέσαμε τις πράξεις για να βρούμε τις συντεταγμένες του σχήματος B-A ως εξής :

(B.x – A.x , B.y – A.y , B.z – A.z)

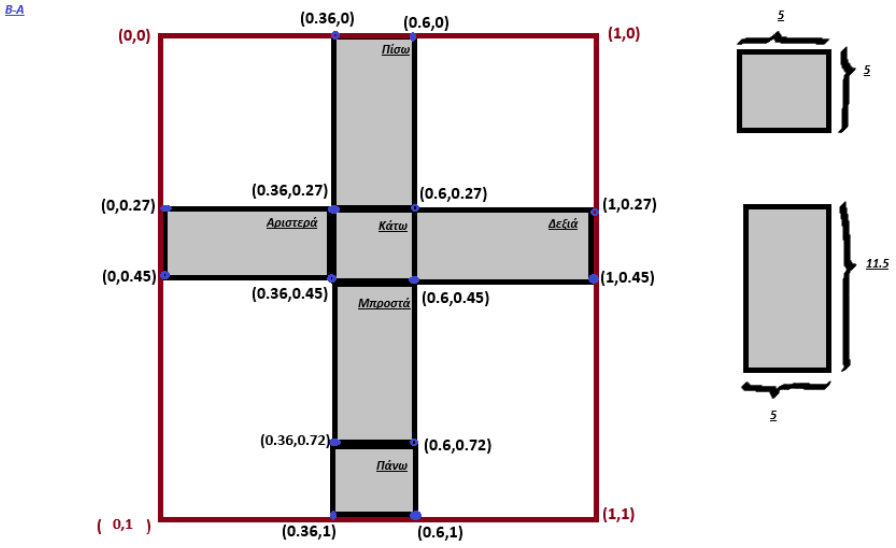


Φτιάξαμε ένα buffer με τις συντεταγμένες για το σχήμα :

```
static const GLfloat DIFF2[] = {

    -2.5f, 14.0f, 5.0f,  //mprosta
     2.5f, 14.0f, 5.0f,
     2.5f, 2.5f, 5.0f,
    2.5f, 2.5f, 5.0f,
    -2.5f, 2.5f, 5.0f,
    -2.5f, 14.0f, 5.0f,
    -2.5f, 14.0f, 0.0f, //dipla aristera
    -2.5f, 14.0f, 5.0f,
    -2.5f, 2.5f, 5.0f,
    -2.5f, 2.5f, 5.0f,
    -2.5f, 2.5f, 0.0f,
    -2.5f, 14.0f, 0.0f,
    2.5f, 14.0f, 0.0f,
    -2.5f, 14.0f, 0.0f,
    -2.5f, 2.5f, 0.0f,
    -2.5f, 2.5f, 0.0f,
    2.5f, 2.5f, 0.0f,
    2.5f, 14.0f, 0.0f,
    2.5f, 14.0f, 5.0f,  //plai dejia
    2.5f, 14.0f, 0.0f,
    2.5f, 2.5f, 0.0f,
    2.5f, 2.5f, 0.0f,
    2.5f, 2.5f, 5.0f,
    2.5f, 14.0f, 5.0f,
    -2.5f, 14.0f, 5.0f,  //panw
    -2.5f, 14.0f, 0.0f,
    2.5f, 14.0f, 0.0f,
    2.5f, 14.0f, 0.0f,
    2.5f, 14.0f, 5.0f,
    -2.5f, 14.0f, 5.0f,
    2.5f, 2.5, 5.0f,
    2.5f, 2.5f, 0.0f,
   -2.5f, 2.5f, 0.0f,
   -2.5f, 2.5f, 0.0f,
   -2.5f, 2.5, 5.0f,
   2.5f, 2.5, 5.0f,

};
```

Unwrapping του σχήματος B-A :

.



Για τον άξονα V αθροίζουμε τις πλευρές 2 τετράγωνων και 2 παραλληλεπίπεδων (5+5+11,5+11,5=33). Για να βρούμε κοινή αναλογία των σχημάτων και να υπολογίσουμε τις κορυφές χωρίσαμε το τετράγωνο ώστε η κάθε πλευρά να είναι 2,5 και αντίστοιχα το

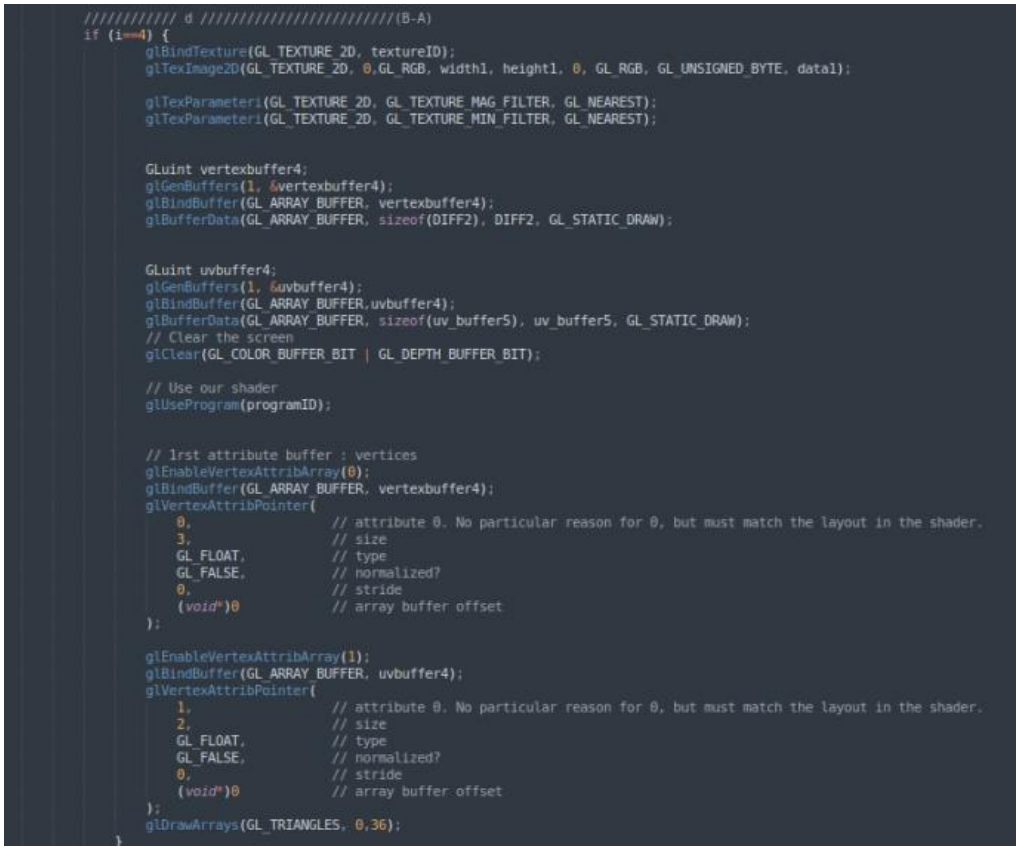
παραλληλεπίπεδο σε 4 ώστε ξανά η πλευρά να είναι 2,5.Οπότε διαιρούμε το 1 με το πλήθος των κορυφών (11) και άρα οι κορυφές απέχουν μεταξύ τους κατά 0,09.

Για τον άξονα U αντίστοιχα έγινε η ιδιά διαδικασία με πάνω . Διαιρούμε το 1 με το πλήθος των κορυφών (9) και άρα οι κορυφές απέχουν μεταξύ τους κατά 0,12.

<u>Ακολουθούν οι συντεταγμένες UV που τοποθετήθηκαν μέσα στον κώδικα μας :</u>

```
static const GLfloat uv_buffer5[] = {

    0.36f, 0.72f,  //mprosta
    0.6f, 0.72f,
    0.6f, 0.45f,
    0.6f, 0.45f,
    0.36f, 0.45f,
    0.36f, 0.72f,
    0.0f, 0.27f,    //plai aristera
    0.36f, 0.27f,
    0.36f, 0.45f,
    0.36f, 0.45f,
    0.0, 0.45f,
    0.0f, 0.27f,
    0.36f, 0.0f,   //pisw
    0.6f, 0.0f,
    0.6f, 0.27f,
    0.6f, 0.27f,
    0.36f, 0.27f,
    0.36f, 0.0f,
    0.6f, 0.27f,    //plai dejia
    1.0f, 0.27f,
    1.0f, 0.45f,
    1.0f, 0.45f,
    0.6, 0.45f,
    0.6f, 0.27f,
    0.6f, 0.72f, //panw
    0.6f, 1.0f,
    0.36f, 1.0f,
    0.36f, 1.0f,
    0.36f, 0.72f,
    0.6f, 0.72f,
    0.6f, 0.27f,  //katw
    0.6f, 0.45f,
    0.36f, 0.45f,
    0.36f, 0.45f,
    0.36f, 0.27f,
    0.6f, 0.27f,

};
```

εφαρμόζουμε στο σχήμα την εικόνα Β.



```
////////// d ////////////////////////(B-A)
if (i==4) {
        glBindTexture(GL_TEXTURE_2D, textureID);
        glTexImage2D(GL_TEXTURE_2D, 0,GL_RGB, width1, height1, 0, GL_RGB, GL_UNSIGNED_BYTE, data1);

        glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST);
        glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);


        GLuint vertexbuffer4;
        glGenBuffers(1, &vertexbuffer4);
        glBindBuffer(GL_ARRAY_BUFFER, vertexbuffer4);
        glBufferData(GL_ARRAY_BUFFER, sizeof(DIFF2), DIFF2, GL_STATIC_DRAW);


        GLuint uvbuffer4;
        glGenBuffers(1, &uvbuffer4);
        glBindBuffer(GL_ARRAY_BUFFER,uvbuffer4);
        glBufferData(GL_ARRAY_BUFFER, sizeof(uv_buffer5), uv_buffer5, GL_STATIC_DRAW);
        // Clear the screen
        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

        // Use our shader
        glUseProgram(programID);


        // 1rst attribute buffer : vertices
        glEnableVertexAttribArray(0);
        glBindBuffer(GL_ARRAY_BUFFER, vertexbuffer4);
        glVertexAttribPointer(
            0,                  // attribute 0. No particular reason for 0, but must match the layout in the shader.
            3,                  // size
            GL_FLOAT,           // type
            GL_FALSE,           // normalized?
            0,                  // stride
            (void*)0            // array buffer offset
        );

        glEnableVertexAttribArray(1);
        glBindBuffer(GL_ARRAY_BUFFER, uvbuffer4);
        glVertexAttribPointer(
            1,                  // attribute 0. No particular reason for 0, but must match the layout in the shader.
            2,                  // size
            GL_FLOAT,           // type
            GL_FALSE,           // normalized?
            0,                  // stride
            (void*)0            // array buffer offset
        );
        glDrawArrays(GL_TRIANGLES, 0,36);
    }
```

v. Η κάμερα θα κινείται στους άξονες του παγκόσμιου συστήματος συντεταγμένων με τους εξής τρόπους:

-γύρω από τον άξονα x με τα πλήκτρα <**w>**και <**x>**

-γύρω από τον άξονα y με τα πλήκτρα <**q**> και <**z**>

-θα κάνει zoom in/zoom out με κατεύθυνση το σημείο Ο με τα πλήκτρα <**+**> και <**-**>του numerical keypad του πληκτρολογίου

```
152
153  void camera_function()
154  {
155      if (glfwGetKey(window, GLFW_KEY_W) == GLFW_PRESS){
156          float rotationn = 0.01;
157          rotationn+=0.01;
158          Model = glm::rotate(Model,rotationn, glm::vec3(rotationn,0.0f,0.0f));
159      }
160      if (glfwGetKey(window, GLFW_KEY_X) == GLFW_PRESS){
161          float rotationn = 0.1;
162          rotationn-=0.15;
163          Model = glm::rotate(Model,rotationn, glm::vec3(-rotationn,0.0f,0.0f));
164      }
165      if (glfwGetKey(window, GLFW_KEY_Q) == GLFW_PRESS){
166          float rotationn2 = 0.01;
167          rotationn2+=0.01;
168          Model = glm::rotate(Model,rotationn2, glm::vec3(0.0f,0.0f,rotationn2));
169      }
170      if (glfwGetKey(window, GLFW_KEY_Z) == GLFW_PRESS){
171          float rotationn2 =0.1;
172          rotationn2-=0.15;
173          Model = glm::rotate(Model,rotationn2, glm::vec3(0.0f,0.0f,-rotationn2));
174      }
175      if (glfwGetKey( window,GLFW_KEY_KP_ADD) == GLFW_PRESS){
176          zoom1-=0.2;
177      }
178      if (glfwGetKey( window, GLFW_KEY_KP_SUBTRACT) == GLFW_PRESS){
179          zoom1+=0.1;
180      }
181          ViewNEW = glm::lookAt(
182          glm::vec3(0.0f,zoom1, 0.0f),
183          glm::vec3(0.0f, 0.0f, 0.0f),
184          glm::vec3(0.0f, 0.0f, 1.0f)
185      );
186  }
187
188
189
```

Τοποθετήσαμε την κάμερα αρχικά στο σημείο 0 .0, 0.0, 40) ώστε να κοιτάει προς το σημείο Ε(0,0,2.5) με ανιόν διάνυσμα (up vector) το (0.0, 1 .0, 0).

```
245
246      // Projection matrix : 45° Field of View, 4:3 ratio, display range : 0.1 unit <-> 100 units
247      glm::mat4 Projection = glm::perspective(glm::radians(45.0f), 4.0f / 4.0f, 0.1f, 100.0f);
248      // Camera matrix
249      glm::mat4 View = glm::lookAt(
250          glm::vec3(0.0f, zoom1, 40.0f), // Camera is at (4,3,-3), in World Space
251          glm::vec3(0.0f, 0.0f, 2.5f), // and looks at the origin
252          glm::vec3(0.0f, 1.0f, 0.0f)  // Head is up (set to 0,-1,0 to look upside-down)
253      );
```

2. Πληροφορίες σχετικά με την υλοποίηση:

Λειτουργικό Σύστημα: Linux

Text Editor : Sublime Text

3. Αξιολόγηση Ομάδας:

Η εργασία εκπονήθηκε από δύο άτομα, όπου ο καθένας έδινε ιδέες για την σχεδίαση της εφαρμογής. Η συνεργασία ήταν άψογη και παρουσιάστηκε συνέπεια ως προς την υλοποίηση της άσκησης.

4. Αναφορές – Πηγές που χρησιμοποιήθηκαν κατά την εκπόνηση της εργασίας.

https://www.geogebra.org/

MeshLab

https://www.youtube.com/watch?v=hePHlQSTii4

https://www.youtube.com/watch?v=Yx2JNbv8Kpg&t=5s