Γραφικά Υπολογιστών και Συστήματα Αλληλεπίδρασης

Γ Μέρος -Σύνολο Προγραμματιστικών Ασκήσεων

Μέλη ομάδας :

Μανίκας Ελευθέριος Μάριος ,4723

Φωτόπουλος Στέφανος ,4829

Ιωάννινα,03/12/2022

Contents

1.Περιγραφή της εργασίας

Ερώτημα 1

Σε αυτήν την γραμμή δημιουργούμε ένα καινούργιο παράθυρο, συγκεκριμένα τετράγωνο με διαστάσεις 1000*1000 και όνομα **«Εργασία 1Γ-Καταστροφή».** 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 1Γ-Καταστροφή", NULL, NULL);
```

Εικόνα 1

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).Μας ζητείται το background να είναι μαύρο οπότε κάθε μια από τις τρεις τιμές θα είναι 0.0f.

```
// Black background
glClearColor(0.0f, 0.0f, 0.0f, 0.0f);
```

Εικόνα 2

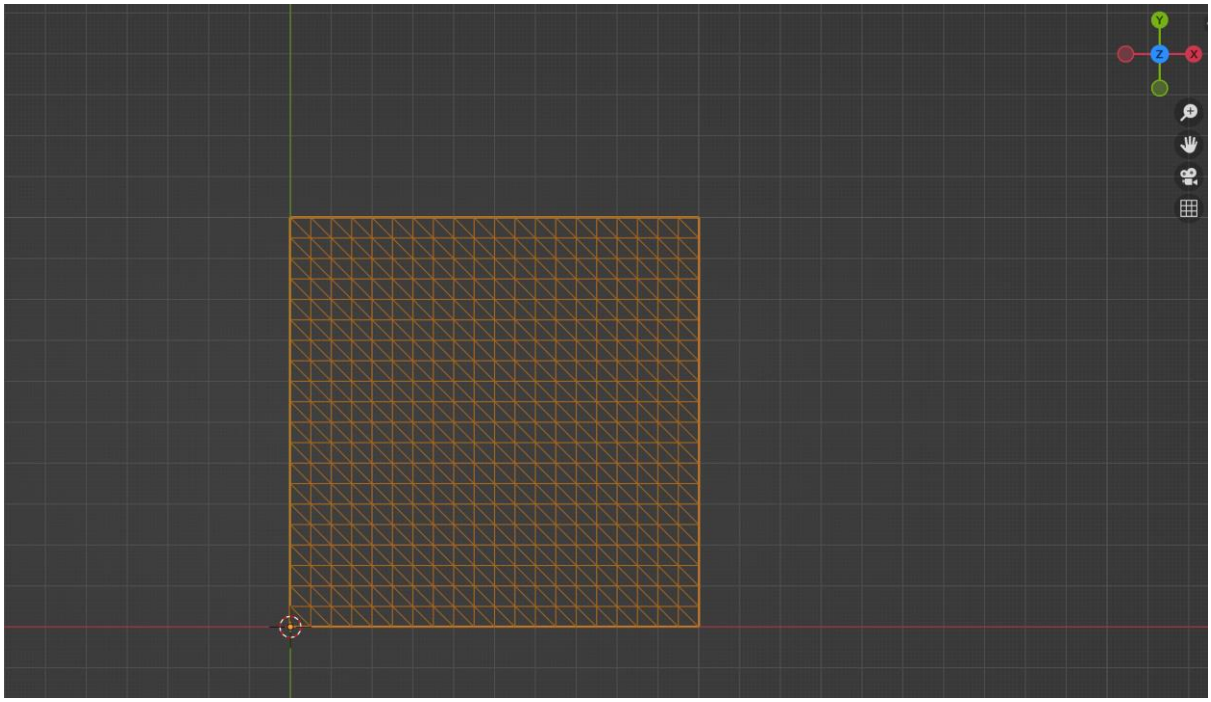
Μέσα σε αυτήν την του while γίνεται ο έλεγχος για τον τερματισμό του προγράμματος. Χρησιμοποιήσαμε την εντολή GLFW_KEY_SPACE, έτσι ώστε ο τερματισμός να γίνεται αν πατήσουμε space όταν είναι ενεργό το παράθυρο η αν κλείσει από το X.

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

Εικόνα 3

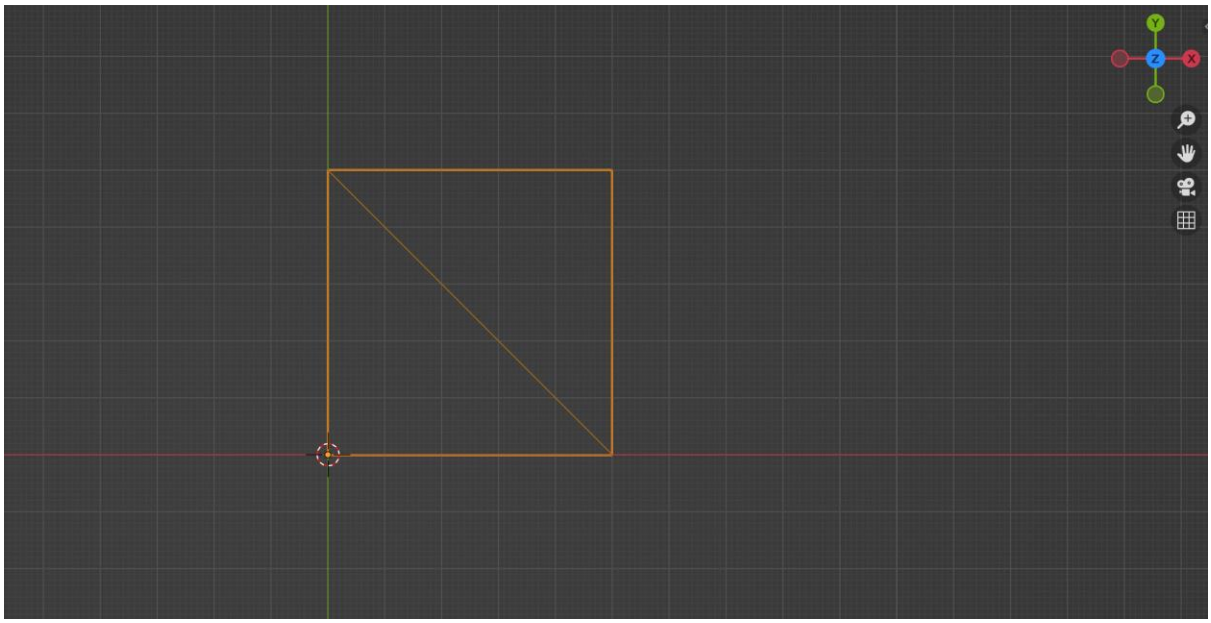
Ερώτημα 2

Το grid το σχεδιάσαμε στο blender με διαστάσεις 20*20(x subdivision*y subdivision) με size 100. Έπειτα στο Transform -> Location X = 50,Location Y = 50.Αυτο το κάναμε για να πετύχουμε την μια γωνιά του grid στο (0,0,0) και οι άξονες να εκτείνονται προς το θετικό x και y.Στο export πατήσαμε την επιλογή triangulated mesh για να σχεδιαστεί το grid με την χρήση τριγώνων(glDrawArrays(GL_TRIANGLES, 0, vertices.size());)

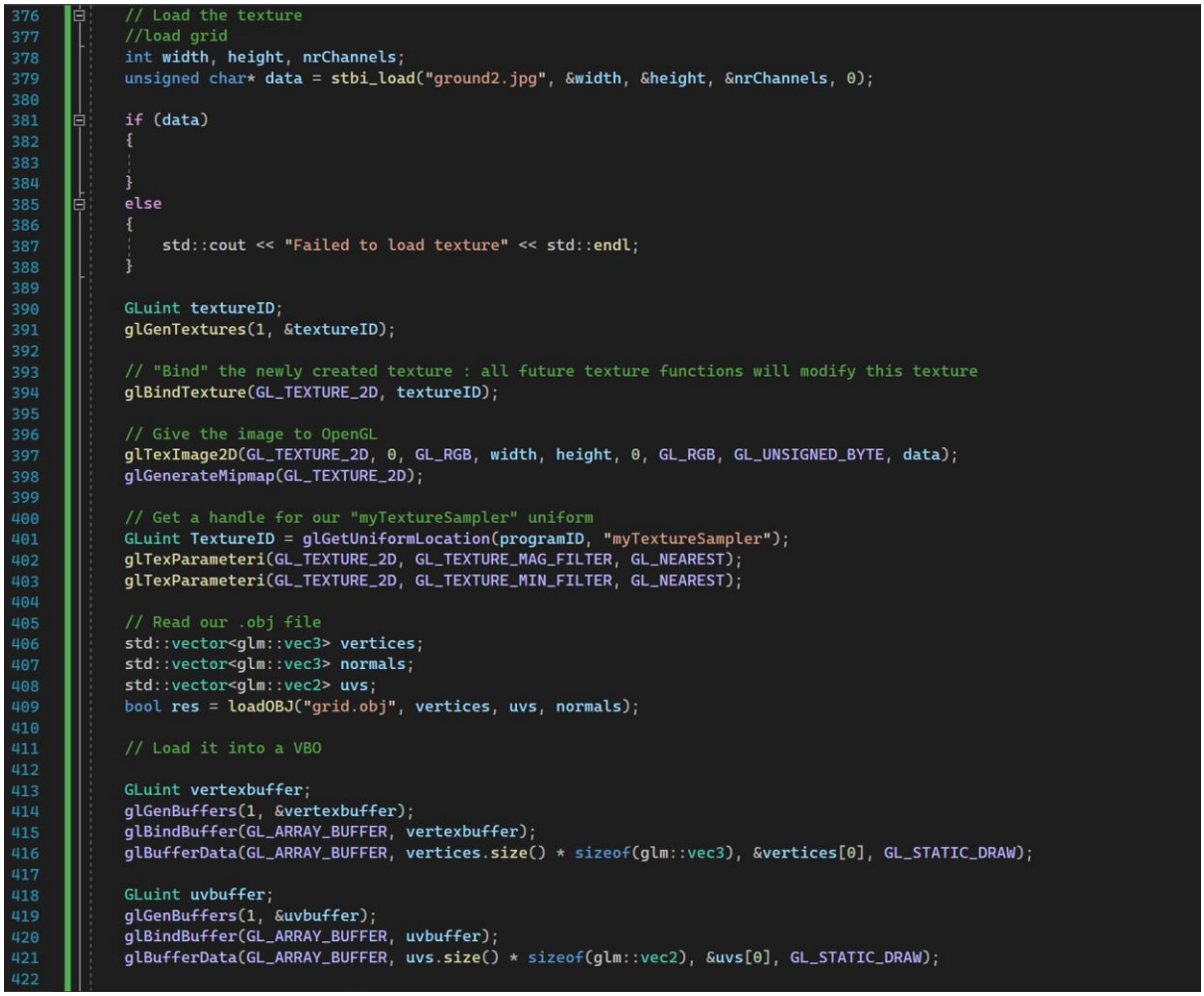


Εικόνα 4

Το παρακάτω grid απεικονίζει το αρχείο gridkratira.obj όπου έχουμε βάλει ένα πλέγμα 1x1 με size=5 ,δηλαδή όσο το μέγεθος ενός κουτιού του αρχικού πλέγματος.



Εικόνα 5



```
// Load the texture
//load grid
int width, height, nrChannels;
unsigned char* data = stbi_load("ground2.jpg", &width, &height, &nrChannels, 0);

if (data)
{

}
else
{
    std::cout << "Failed to load texture" << std::endl;
}

GLuint textureID;
glGenTextures(1, &textureID);

// "Bind" the newly created texture : all future texture functions will modify this texture
glBindTexture(GL_TEXTURE_2D, textureID);

// Give the image to OpenGL
glTexImage2D(GL_TEXTURE_2D, 0, GL_RGB, width, height, 0, GL_RGB, GL_UNSIGNED_BYTE, data);
glGenerateMipmap(GL_TEXTURE_2D);

// Get a handle for our "myTextureSampler" uniform
GLuint TextureID = glGetUniformLocation(programID, "myTextureSampler");
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);

// Read our .obj file
std::vector<glm::vec3> vertices;
std::vector<glm::vec3> normals;
std::vector<glm::vec2> uvs;
bool res = loadOBJ("grid.obj", vertices, uvs, normals);

// Load it into a VBO

GLuint vertexbuffer;
glGenBuffers(1, &vertexbuffer);
glBindBuffer(GL_ARRAY_BUFFER, vertexbuffer);
glBufferData(GL_ARRAY_BUFFER, vertices.size() * sizeof(glm::vec3), &vertices[0], GL_STATIC_DRAW);

GLuint uvbuffer;
glGenBuffers(1, &uvbuffer);
glBindBuffer(GL_ARRAY_BUFFER, uvbuffer);
glBufferData(GL_ARRAY_BUFFER, uvs.size() * sizeof(glm::vec2), &uvs[0], GL_STATIC_DRAW);

```

Εικόνα 6

Στην παραπάνω εικόνα βλέπουμε την φόρτωση του grid.obj και δημιουργούμε vertexbuffer,uvbuffer για το grid μας τα οποία θα χρησιμοποιηθούν αργότερα στο bind.

Ερώτημα 3

Στην εικόνα 7 βλέπουμε την τοποθεσία της κάμερας που έχουμε θέσει, το κέντρο πλέγματος είναι το target της(σε αυτό γίνεται το zoom in/zoom out) και το ανιόν διάνυσμα με 1.0 στον άξονα z.

```
    glm::mat4 Projection = glm::perspective(glm::radians(45.0f), float(1000 / 1000), 0.1f, 10000.0f);
    // Camera matrix
    glm::mat4 View = glm::lookAt(
        glm::vec3(-20.0f, -20.0f, 40.0f),//position
        glm::vec3(50.0f, 50.0f, 0.0f),// target
        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()
```

Εικόνα 7

Κρατήσαμε την ίδια λειτουργία κάμερας με κάποιες αλλαγές όπως μας ζητείται. Συγκεκριμένα η κάμερα κινείται πλέον γύρω από τον άξονα z με τα πλήκτρα και <a> και <d>. Επίσης αλλάξαμε στο Projection το FAR CLIPPING PLANE στο 10000 για να μην κάνει clipping το grid.

```
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 D  Dexiostrofa
    if (glfwGetKey(window, GLFW_KEY_D) == GLFW_PRESS) {
        ViewMatrix = glm::rotate(ViewMatrix, radians(1.0f), glm::vec3(0.0f, 0.0f, 1.0f));
    }
    //STRAFE STON AXONA A Aristerostrofa
    if (glfwGetKey(window, GLFW_KEY_A) == 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 - 0.7;
        if (zoom)
        {
            ProjectionMatrix = glm::perspective(glm::radians(zoom), float(1000 / 1000), 0.1f, 10000.0f);
        }
    }
    //ZOOM OUT
    if (glfwGetKey(window, GLFW_KEY_KP_SUBTRACT) == GLFW_PRESS)
    {
        zoom = zoom + 0.7;
        if (zoom)
        {
            ProjectionMatrix = glm::perspective(glm::radians(zoom), float(1000 / 1000), 0.1f, 10000.0f);
        }
    }
}
```

Εικόνα 8

Ερώτημα 4

Αρχικά κάνουμε Load το object της σφαίρας που μας έχει δοθεί δημιουργώντας παράλληλα vertexbufferball,uvbufferball για την σφαίρα τα οποία θα χρησιμοποιηθούν αργότερα στο bind.

```
444     //load sfairas
445     int widthball, heightball, nrChannelsball;
446     unsigned char* data1 = stbi_load("fire.jpg", &widthball, &heightball, &nrChannelsball, 0);
447
448     if (data1)
449     {
450
451     }
452     else
453     {
454         std::cout << "Failed to load texture" << std::endl;
455     }
456
457     GLuint textureIDball;
458     glGenTextures(1, &textureIDball);
459
460     // "Bind" the newly created texture : all future texture functions will modify this texture
461     glBindTexture(GL_TEXTURE_2D, textureIDball);
462
463     // Give the image to OpenGL
464     glTexImage2D(GL_TEXTURE_2D, 0, GL_RGB, widthball, heightball, 0, GL_RGB, GL_UNSIGNED_BYTE, data1);
465     glGenerateMipmap(GL_TEXTURE_2D);
466
467     // Get a handle for our "myTextureSampler" uniform
468     GLuint TextureIDball = glGetUniformLocation(programID, "myTextureSampler");
469     glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST);
470     glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);
471
472     // Read our .obj file
473     std::vector<glm::vec3> verticesball;
474     std::vector<glm::vec3> normalsball;
475     std::vector<glm::vec2> uvsball;
476     bool resball = loadOBJ("ball.obj", verticesball, uvsball, normalsball);
477
478     GLuint vertexbufferball;
479     glGenBuffers(1, &vertexbufferball);
480     glBindBuffer(GL_ARRAY_BUFFER, vertexbufferball);
481     glBufferData(GL_ARRAY_BUFFER, verticesball.size() * sizeof(glm::vec3), &verticesball[0], GL_STATIC_DRAW);
482
483     GLuint uvbufferball;
484     glGenBuffers(1, &uvbufferball);
485     glBindBuffer(GL_ARRAY_BUFFER, uvbufferball);
486     glBufferData(GL_ARRAY_BUFFER, uvsball.size() * sizeof(glm::vec2), &uvsball[0], GL_STATIC_DRAW);
```

Εικόνα 9

Φτιάξαμε μια συνάρτηση Random η οποία με κάθε κάλεσμά της μας δίνει τυχαίες τιμές για τον άξονα x και τον άξονα y με σταθερή τιμή στον άξονα z =20.Το range που παίρνει τυχαίες τιμές είναι το (0,100) όπως άλλωστε ζητείται και από την εκφώνηση.

```
52  float Random(float* x_axis, float* y_axis, float* z_axis) {
53      srand(time(0));
54      *x_axis = 0 + (rand() % 100);
55      *y_axis = 0 + (rand() % 100);
56      *z_axis = 20;
57
58      cout << *x_axis << "\n";
59      cout << *y_axis << "\n";
60      return 0;
61  }
```

Εικόνα 10

Αρχικοποιούμε έξω από την do τα x,y,z και την μεταβλητή flag η οποία έχει την εξής λειτουργία:

Όταν ο χρήστης πατάει το B παίρνει την τιμή 1 και όταν η μπάλα φτάσει στο έδαφος παίρνει την τιμή 0.

Έπειτα δημιουργούμε μια struct στην οποία θα έχουμε τους άξονες x,y και φτιάχνουμε έναν πίνακα με τον τύπο της struct ο οποίος θα έχει max τιμή το 400 όσο και τα τετράγωνα του grid.

Οι μεταβλητές thesi_pinaka και counter_of_craters θα εξηγηθούν παρακάτω.

```
535     int flag = 0;
536     float x_axis = 0;
537     float y_axis = 0;
538     float z_axis = 20;
539
540
541     struct Coordinates
542     {
543         int x_axis;
544         int y_axis;
545     };
546     struct Coordinates save_crater[400];
547     int thesi_pinaka = 0;
548     int counter_of_craters = 0;
```

Εικόνα 11

Στην εικόνα 12 κάνουμε Bind με τα Buffers που έχουμε προαναφέρει για το grid και το ζωγραφίζουμε .

```
569         // Bind our texture in Texture Unit 0
570         glActiveTexture(GL_TEXTURE0);
571         glBindTexture(GL_TEXTURE_2D, textureID);
572         // Set our "myTextureSampler" sampler to use Texture Unit 0
573         glUniform1i(TextureID, 0);
574
575         // 1rst attribute buffer : vertices
576         glEnableVertexAttribArray(0);
577         glBindBuffer(GL_ARRAY_BUFFER, vertexbuffer);
578         glVertexAttribPointer(
579             0,                  // attribute
580             3,                  // size
581             GL_FLOAT,           // type
582             GL_FALSE,           // normalized?
583             0,                  // stride
584             (void*)0            // array buffer offset
585         );
586
587         // 2nd attribute buffer : UVs
588         glEnableVertexAttribArray(1);
589         glBindBuffer(GL_ARRAY_BUFFER, uvbuffer);
590         glVertexAttribPointer(
591             1,                                // attribute
592             2,                                // size
593             GL_FLOAT,                         // type
594             GL_FALSE,                         // normalized?
595             0,                                // stride
596             (void*)0                          // array buffer offset
597         );
598
599         // Draw the triangle !
600         glDrawArrays(GL_TRIANGLES, 0, vertices.size());
```

Εικόνα 12

Αν πατηθεί το πλήκτρο B και το flag που ορίσαμε έχει τιμή 0 αλλάζουμε την τιμή αυτού και καλούμε την συνάρτηση Random που δημιουργήσαμε παραπάνω.

```
605         if (glfwGetKey(window, GLFW_KEY_B) == GLFW_PRESS && flag == 0) {
606
607             flag = 1;
608             Random(&x_axis, &y_axis, &z_axis);
609         }
```

Εικόνα 13

Όταν η τιμή του flag μας είναι 1(δηλαδή πατηθεί το B) φτιάχνουμε ένα RandomVector στο οποίο κρατάμε τις τιμές της Random,κάνουμε scale για την μπάλα με 10 σε όλους τους άξονες έτσι ώστε να έχει διάμετρο 5 όσο ένα τετράγωνο του grid. Έπειτα, translate με RandomVector και πολλαπλασιασμός αυτών των δυο με την σωστή σειρά για να πάρουμε το επιθυμητό αποτέλεσμα και MVP για την μπάλα. Ομοίως, γίνεται Bind των buffers της σφαίρας όπως έγινε και για το Grid.

```
610     if (flag == 1) {
611         glm::vec3 RandomVector = glm::vec3(x_axis, y_axis, z_axis);
612         glm::mat4 Modelball = glm::scale(glm::mat4(1.0f), glm::vec3(10.0f, 10.0f, 10.0f));
613         glm::mat4 TranslationMatrix = glm::translate(glm::mat4(1.0f), RandomVector);
614         Modelball = TranslationMatrix * Modelball;
615
616
617
618         glm::mat4 MVPball = Projection * View * Modelball;// ypologizw to kainourgio mvp symfwna me ta kainourgia dedomena
619         glUniformMatrix4fv(MatrixID, 1, GL_FALSE, &MVPball[0][0]);
620         // Load it into a VBO
621
622         // Bind our texture in Texture Unit 0
623         glActiveTexture(GL_TEXTURE0);
624         glBindTexture(GL_TEXTURE_2D, textureIDball);
625         // Set our "myTextureSampler" sampler to use Texture Unit 0
626         glUniform1i(TextureIDball, 0);
627
628         // 1rst attribute buffer : vertices
629         glEnableVertexAttribArray(0);
630         glBindBuffer(GL_ARRAY_BUFFER, vertexbufferball);
631         glVertexAttribPointer(
632             0,                  // attribute
633             3,                  // size
634             GL_FLOAT,           // type
635             GL_FALSE,           // normalized?
636             0,                  // stride
637             (void*)0            // array buffer offset
638         );
639
640         // 2nd attribute buffer : UVs
641         glEnableVertexAttribArray(1);
642         glBindBuffer(GL_ARRAY_BUFFER, uvbufferball);
643         glVertexAttribPointer(
644             1,                                // attribute
645             2,                                // size
646             GL_FLOAT,                         // type
647             GL_FALSE,                         // normalized?
648             0,                                // stride
649             (void*)0                          // array buffer offset
650         );
651
```

Εικόνα 14

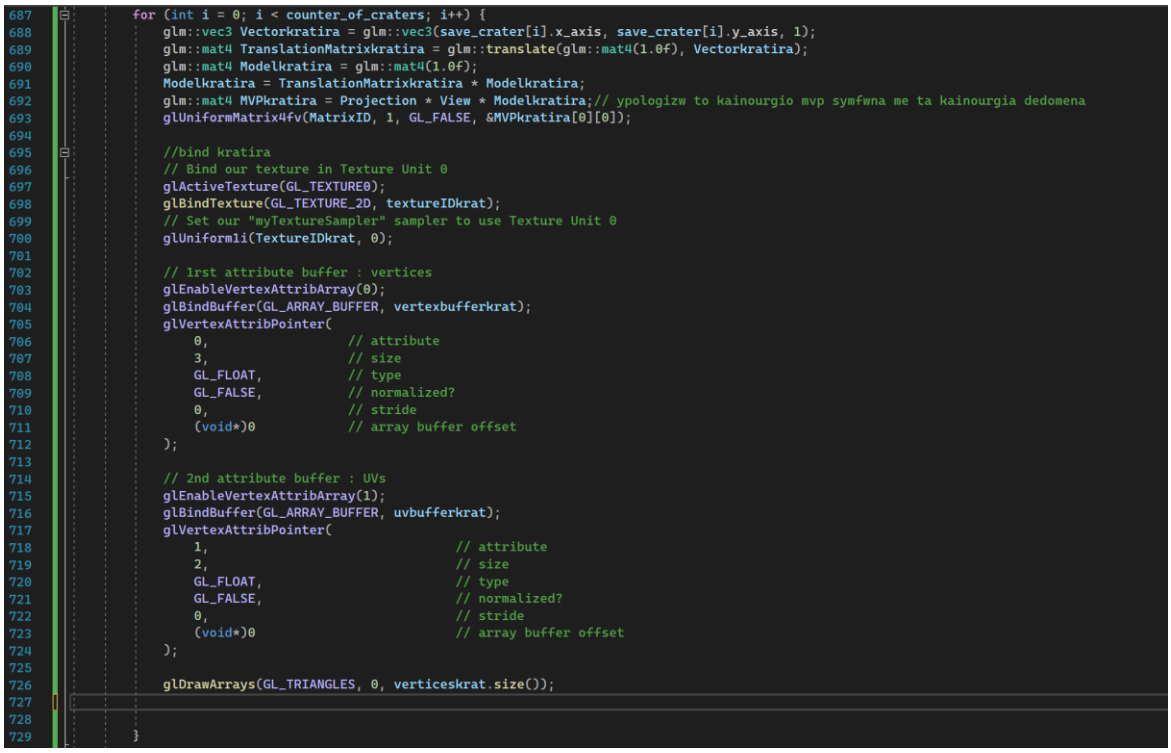
BONUS 1&2

Μέσα στο if(flag==1) εισάγουμε και το παρακάτω if, η λειτουργία του οποίου είναι να μειώνει τον άξονα z όσο η σφαίρα δεν έχει φτάσει στο έδαφος με ταχύτητα -0.2.Για αυτό κάνουμε translate τον μοναδιαίο πίνακα με ένα vector με -0.2 στο τρίτο όρισμα. Μετά την δημιουργία του νέου MVP παρατηρούμε δυο ακόμη εμφωλιασμένα if που αναφέρονται στο(Bonus β)).Όπου στο πρώτο αφαιρούμε ακομή 0.5 στον άξονα z με αποτέλεσμα να πέφτει πιο γρήγορα η μπάλα και στο δεύτερο προσθέτουμε αρχικά 0.1 στον άξονα z αλλά μετά στο translate αφαιρούμε την τιμή 0.1 με αυτόν τον τρόπο μειώνουμε την ταχύτητα της καθόδου της σφαίρας. Έπειτα, όταν η σφαίρα αγγίξει το 0 στον άξονα z δημιουργούμε ήχο με την Playsound που έχει εισαχθεί χάρις στα includes που βάλαμε στην αρχή του κώδικα(`#include <Windows.h>`,`#include <MMSystem.h>`).Αποθηκεύουμε τις τιμές x,y στον save_crater έτσι ώστε να προβάλλονται και οι παλαιότεροι κρατήρες, προσθέτουμε +1 στο counter_of_craters για να γνωρίζουμε το πλήθος των κρατήρων, προσθέτουμε +1 στην θέση του πίνακα με σκοπό να γράψει στην επόμενη γραμμή του πίνακα save_crater αν ξαναμπεί σε αυτό το If.Τέλος, αλλάζουμε την τιμή του flag σε 0 αφού η σφαίρα έχει φτάσει πλέον το έδαφος και περιμένουμε από τον χρήστη το επόμενο πάτημα του πλήκτρου B.

```
652
653             if (z_axis > 0.0) {
654                 z_axis -= 0.2;
655                 glm::mat4 TranslationMatrix1 = glm::translate(glm::mat4(1.0f), glm::vec3(0.0f, 0.0f, -0.2f));
656                 Modelball = TranslationMatrix1 * Modelball;
657                 MVPball = Projection * View * Modelball;// ypologizw to kainourgio mvp symfwna me ta kainourgia dedomena
658                 if (glfwGetKey(window, GLFW_KEY_U) == GLFW_PRESS) {
659                     z_axis = z_axis - 0.5;
660                     glm::mat4 TranslationMatrix1 = glm::translate(glm::mat4(1.0f), glm::vec3(0.0f, 0.0f, -0.5f));
661                     Modelball = TranslationMatrix1 * Modelball;
662                     MVPball = Projection * View * Modelball;
663                 }
664                 if (glfwGetKey(window, GLFW_KEY_P) == GLFW_PRESS) {
665                     z_axis = z_axis + 0.1;
666                     glm::mat4 TranslationMatrix1 = glm::translate(glm::mat4(1.0f), glm::vec3(0.0f, 0.0f, -0.1f));
667                     Modelball = TranslationMatrix1 * Modelball;
668                     MVPball = Projection * View * Modelball;
669                 }
670                 //glUniformMatrix4fv(MatrixID, 1, GL_FALSE, &MVPball[0][0]);
671                 glDrawArrays(GL_TRIANGLES, 0, verticesball.size());
672                 glDisableVertexAttribArray(0);
673                 glDisableVertexAttribArray(1);
674
675                 if (z_axis <= 0.0) {
676                     PlaySound(TEXT("thunder2.wav"), NULL, SND_ASYNC | SND_FILENAME);
677                     save_crater[thesi_pinaka].x_axis = x_axis;
678                     save_crater[thesi_pinaka].y_axis = y_axis;
679                     counter_of_craters += 1;
680                     thesi_pinaka += 1;
681                     flag = 0;
682
683                 }
684             }
685         }
```

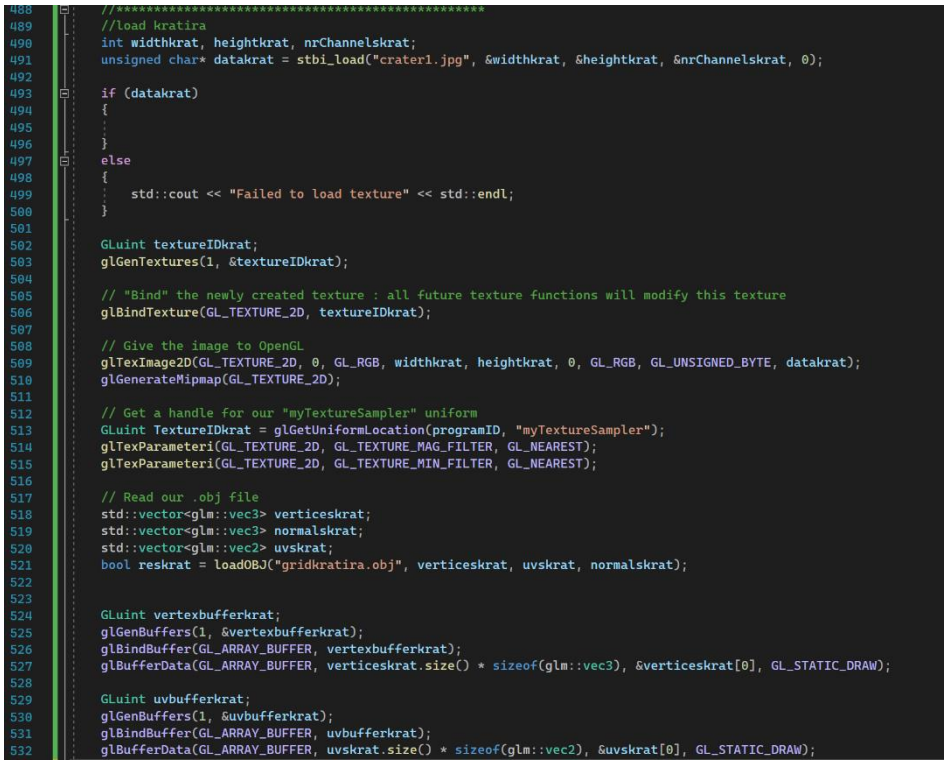
Εικόνα 15

Αφού έχουμε βγει έξω από τα if της σφαίρας τώρα θα σχεδιάσουμε τους κρατήρες αυτό γίνεται με μια for όσο το πλήθος των κρατήρων. Στην συνέχεια, δημιουργούμε Vectorkratira το οποίο έχει σαν τιμές την x τιμή του save_crater για τον κάθε κρατήρα ομοίως για το y,σαν τρίτη τιμή βλέπουμε 1 αυτό έχει γίνει διότι με 0 ο κρατήρας μας εμφανιζόταν επάνω στο grid και είχαμε επικάλυψη. Τέλος ,γίνεται ο υπολογισμός του καινούριο MVP και κάνουμε bind τα texture του κρατήρα με τους buffer που έχουμε δημιουργήσει στην εικόνα 16.



```
for (int i = 0; i < counter_of_craters; i++) {
    glm::vec3 Vectorkratira = glm::vec3(save_crater[i].x_axis, save_crater[i].y_axis, 1);
    glm::mat4 TranslationMatrixkratira = glm::translate(glm::mat4(1.0f), Vectorkratira);
    glm::mat4 Modelkratira = glm::mat4(1.0f);
    Modelkratira = TranslationMatrixkratira * Modelkratira;
    glm::mat4 MVPkratira = Projection * View * Modelkratira;// ypologizw to kainourgio mvp symfwna me ta kainourgia dedomena
    glUniformMatrix4fv(MatrixID, 1, GL_FALSE, &MVPkratira[0][0]);

    //bind kratira
    // Bind our texture in Texture Unit 0
    glActiveTexture(GL_TEXTURE0);
    glBindTexture(GL_TEXTURE_2D, textureIDkrat);
    // Set our "myTextureSampler" sampler to use Texture Unit 0
    glUniform1i(TextureIDkrat, 0);

    // 1rst attribute buffer : vertices
    glEnableVertexAttribArray(0);
    glBindBuffer(GL_ARRAY_BUFFER, vertexbufferkrat);
    glVertexAttribPointer(
        0,                  // attribute
        3,                  // size
        GL_FLOAT,           // type
        GL_FALSE,           // normalized?
        0,                  // stride
        (void*)0            // array buffer offset
    );

    // 2nd attribute buffer : UVs
    glEnableVertexAttribArray(1);
    glBindBuffer(GL_ARRAY_BUFFER, uvbufferkrat);
    glVertexAttribPointer(
        1,                                // attribute
        2,                                // size
        GL_FLOAT,                         // type
        GL_FALSE,                         // normalized?
        0,                                // stride
        (void*)0                          // array buffer offset
    );

    glDrawArrays(GL_TRIANGLES, 0, verticeskrat.size());


}
```

Εικόνα 16



```
//*********************************************
//load kratira
int widthkrat, heightkrat, nrChannelskrat;
unsigned char* datakrat = stbi_load("crater1.jpg", &widthkrat, &heightkrat, &nrChannelskrat, 0);

if (datakrat)
{


}
else
{
    std::cout << "Failed to load texture" << std::endl;
}

GLuint textureIDkrat;
glGenTextures(1, &textureIDkrat);

// "Bind" the newly created texture : all future texture functions will modify this texture
glBindTexture(GL_TEXTURE_2D, textureIDkrat);

// Give the image to OpenGL
glTexImage2D(GL_TEXTURE_2D, 0, GL_RGB, widthkrat, heightkrat, 0, GL_RGB, GL_UNSIGNED_BYTE, datakrat);
glGenerateMipmap(GL_TEXTURE_2D);

// Get a handle for our "myTextureSampler" uniform
GLuint TextureIDkrat = glGetUniformLocation(programID, "myTextureSampler");
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);

// Read our .obj file
std::vector<glm::vec3> verticeskrat;
std::vector<glm::vec3> normalskrat;
std::vector<glm::vec2> uvskrat;
bool reskrat = loadOBJ("gridkratira.obj", verticeskrat, uvskrat, normalskrat);


GLuint vertexbufferkrat;
glGenBuffers(1, &vertexbufferkrat);
glBindBuffer(GL_ARRAY_BUFFER, vertexbufferkrat);
glBufferData(GL_ARRAY_BUFFER, verticeskrat.size() * sizeof(glm::vec3), &verticeskrat[0], GL_STATIC_DRAW);

GLuint uvbufferkrat;
glGenBuffers(1, &uvbufferkrat);
glBindBuffer(GL_ARRAY_BUFFER, uvbufferkrat);
glBufferData(GL_ARRAY_BUFFER, uvskrat.size() * sizeof(glm::vec2), &uvskrat[0], GL_STATIC_DRAW);
```

Εικόνα 17

Τέλος, κάνουμε delete κάθε buffer και texture που έχουμε δημιουργήσει.

```
744     // Check if the SPACE key was pressed or the window was closed
745     while (glfwGetKey(window, GLFW_KEY_SPACE) != GLFW_PRESS &&
746         glfwWindowShouldClose(window) == 0);
747
748     // Cleanup VBO and shader
749     glDeleteBuffers(1, &vertexbuffer);
750     glDeleteBuffers(1, &uvbuffer);
751     glDeleteBuffers(1, &vertexbufferball);
752     glDeleteBuffers(1, &uvbufferball);
753     glDeleteBuffers(1, &vertexbufferkrat);
754     glDeleteBuffers(1, &uvbufferkrat);
755     glDeleteProgram(programID);
756     glDeleteTextures(1, &textureID);
757     glDeleteTextures(1, &textureIDball);
758     glDeleteTextures(1, &textureIDkrat);
759     glDeleteVertexArrays(1, &VertexArrayID);
760
761     // Close OpenGL window and terminate GLFW
762     glfwTerminate();
763
764     return 0;
765
766 }
```

Εικόνα 18

2.Πληροφορίες σχετικά με την υλοποίηση

Η υλοποίηση της άσκησης έγινε στο λειτουργικό σύστημα Windows 10 και χρησιμοποιώντας το πρόγραμμα Visual Studio x64. Για τον ήχο θα χρειαστεί να πάτε στο properties του project->Linker->Input->AdditionalDependencies και να εισάγετε **opengl32.lib;glfw3.lib;glfw3dll.lib;glew32.lib;winmm.lib;%(AdditionalDependencies)**

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://stackoverflow.com/questions/10468128/how-do-you-make-an-array-of-structs-in-c

https://stackoverflow.com/questions/37598963/how-do-i-have-an-array-that-has-a-set-of-coordinates-in-c

Η ιδέα για τον πίνακα save_crater πάρθηκε από τα παραπάνω links.

https://stackoverflow.com/questions/8804880/use-playsound-in-c-opengl-to-play-sound-in-background

Για την PlaySound.