ΑΣΚΗΣΗ 1-Γ OPENGL

ΣΤΟΙΧΕΙΑ ΟΜΑΔΑΣ: -Τσαπικούνη Γεωργία

-Κοτοφώλη Χριστίνα

ΗΜΕΡΟΜΗΝΙΑ: 4/12/2022

ΠΕΡΙΓΡΑΦΗ ΕΡΓΑΣΙΑΣ

➢ Ερώτημα i

Αρχικά στο πρώτο ερώτημα μας ζητείτε να ανοίγει ένα βασικό παράθυρο 1000x1000 και να εμφανίζεται ως τίτλος «Εργασία 1Γ- Καταστροφή». Αυτό υλοποιείται με την εντολή **‘’window = glfwCreateWindow(1000, 1000, "Ergasia 1C-katastrofi", NULL, NULL);’’**.Το background το κάνουμε μαύρο **‘’ glClearColor(0.0f, 0.0f, 0.0f, 0.0f);’’**.Τέλος , η εφαρμογή τερματίζει πατώντας το πλήκτρο space **‘’(glfwGetKey(window, GLFW_KEY_SPACE) != GLFW_PRESS && glfwWindowShouldClose(window) == 0);’’**.Η ανάλυση αυτού του ερωτήματος πραγματοποιήθηκε στην πρώτη άσκηση.

➢ Ερώτημα ii

Στο συγκεκριμένο ερώτημα έχουμε φορτώσει τρία object μέσω του προγράμματος Blender. Αρχικά, έχουμε δημιουργήσει ένα grid1.obj το πλέγμα του οποίου έχει σχεδιαστεί πάνω στο x,y επίπεδο με z=0 και επεκτείνεται στον άξονα xε(0,200) και στον άξονα yε(0,200). Αποτελείται από 400 τετράγωνα με κέντρο πλέγματος στο (100,100,0). Στο συγκεκριμένο πλέγμα έχουμε εφαρμόσει την εικόνα υφής ground2.jpg. Στο κώδικα μας, όπως φαίνεται στις παρακάτω εικόνες έχουμε δημιουργήσει ένα MVP πίνακα, έχουμε διαβάσει το texture ground2.jpg , grid1.obj και το έχουμε φορτώσει σε ένα VBO.

```
/*************************grid1*************************************/
GLuint VertexArrayID;
glGenVertexArrays(1, &VertexArrayID);
glBindVertexArray(VertexArrayID);
```

```
GLuint MatrixID = glGetUniformLocation(programID, "MVP");/*----------grid1--------*/
```

```
int width, height, nrChannels;
unsigned char* data = stbi_load("ground2.jpg", &width, &height, &nrChannels, 0);

if (data)
{
    std::cout << "Texture loaded succesfully" << std::endl;

}
else {

    std::cout << "Failed to load texture" << std::endl;

}


GLuint textureID;
glGenTextures(1, &textureID);

// Give the Grid image to OpenGL
glBindTexture(GL_TEXTURE_2D, textureID);
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");
```

```
// Give the Grid image to OpenGL
glBindTexture(GL_TEXTURE_2D, textureID);
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");

// Read our grid.obj file
std::vector<glm::vec3> vertices;
std::vector<glm::vec3> normals;
std::vector<glm::vec2> uvs;
bool res = loadOBJ("grid1.obj", vertices, uvs, normals);

if (res)
{
    std::cout << "OBJ loaded succesfully" << std::endl;

}
else {

    std::cout << "Failed to load OBJ" << std::endl;

}

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);
```

- Ερώτημα iii

Αρχικά τοποθετούμε την κάμερα στο σημείο (100,100,-99) , κοιτάει προς το κέντρο του πλέγματος (100,500,-100) με up vector (0,1,0).

```
glm::vec3 positionCam = glm::vec3(100.0f, 100.0f, -99.0f);
glm::vec3 frontDirectionCam = glm::vec3(-1.0f, 0.0f, 0.0f);
glm::vec3 up = glm::vec3(0.0f, 1.0f, 0.0f);

const float speedCam = 0.3f;
void camera_function()
{
    glm::vec3 startCam = glm::vec3(100.0f, 500.0f, -100.0f);
    glm::vec3 direction = glm::normalize(positionCam - startCam);
    ProjectionMatrix = glm::perspective(glm::radians(140.0f), 4.0f / 3.0f, 0.1f, 10000.0f);
    ViewMatrix = glm::lookAt(positionCam, positionCam + direction, up);
```

Εικόνα 1

Σε αυτό το ερώτημα υλοποιούμε την κίνηση της κάμερας γύρω από τον άξονα x και z καθώς και zoom in, zoom out με την χρήση συγκεκριμένων πλήκτρων.

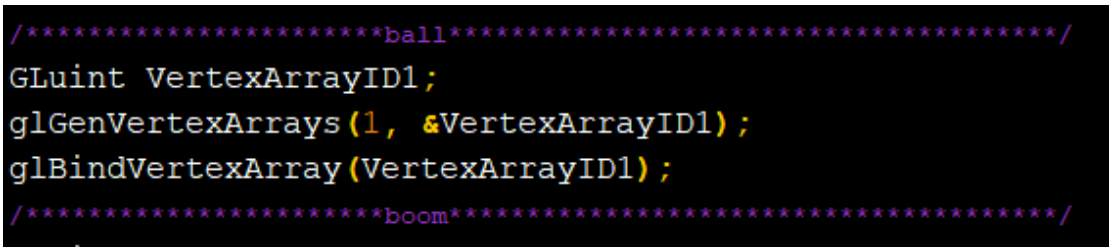
```
glm::vec3 positionCam = glm::vec3(100.0f, 100.0f, -99.0f);
glm::vec3 frontDirectionCam = glm::vec3(-1.0f, 0.0f, 0.0f);
glm::vec3 up = glm::vec3(0.0f, 1.0f, 0.0f);

const float speedCam = 0.3f;
void camera_function()
{
    glm::vec3 startCam = glm::vec3(100.0f, 500.0f, -100.0f);
    glm::vec3 direction = glm::normalize(positionCam - startCam);
    ProjectionMatrix = glm::perspective(glm::radians(140.0f), 4.0f / 3.0f, 0.1f, 10000.0f);
    ViewMatrix = glm::lookAt(positionCam, positionCam + direction, up);


    if (glfwGetKey(window, GLFW_KEY_X) == GLFW_PRESS)
        positionCam += glm::normalize(glm::cross(up, direction)) * speedCam;
    if (glfwGetKey(window, GLFW_KEY_W) == GLFW_PRESS)
        positionCam -= glm::normalize(glm::cross(up, direction)) * speedCam;



    if (glfwGetKey(window, GLFW_KEY_A) == GLFW_PRESS)
        positionCam += glm::normalize(glm::cross(frontDirectionCam, direction)) * speedCam;
    if (glfwGetKey(window, GLFW_KEY_D) == GLFW_PRESS)
        positionCam -= glm::normalize(glm::cross(frontDirectionCam, direction)) * speedCam;



    if (glfwGetKey(window, GLFW_KEY_KP_ADD) == GLFW_PRESS)
        positionCam += speedCam * direction;
    if (glfwGetKey(window, GLFW_KEY_KP_SUBTRACT) == GLFW_PRESS)
        positionCam -= speedCam * direction;


}
```

Εικόνα 2

Κάθε φορά που πατάμε ένα από τα πλήκτρα X,W,A,D,+,-, η θέση της κάμερας ενημερώνεται ανάλογα. Αν θέλουμε να κάνουμε zoom in προς το κέντρο του πλέγματος ή zoom out από αυτό προσθέτουμε ή αφαιρούμε το διάνυσμα κατεύθυνσης από το διάνυσμα θέσης κλιμακούμενο κατά κάποια τιμή ταχύτητας ‘’**positionCam ±= speedCam * direction;’’**. Αν θέλουμε να περιστραφούμε γύρω από τον άξονα x κάνουμε εξωτερικό γινόμενο του up vector με την κατεύθυνση της κάμερας κλιμακούμενο κατά κάποια τιμή ταχύτητας **‘’positionCam += glm::normalize(glm::cross(up, direction)) * speedCam;’’** για να δημιουργήσουμε ένα σωστό διάνυσμα και κινούμαστε γύρω από τον άξονα x . Αντίστοιχα, για να περιστραφούμε γύρω από τον άξονα z κάνουμε εξωτερικό γινόμενο του διανύσματος κατεύθυνσης της κάμερας με το διάνυσμα που κοιτάει πάντα η κάμερα κλιμακούμενο κατά κάποια τιμή ταχύτητας **‘’positionCam += glm::normalize(glm::cross(frontDirectionCam,direction)) * speedCam;’’.**

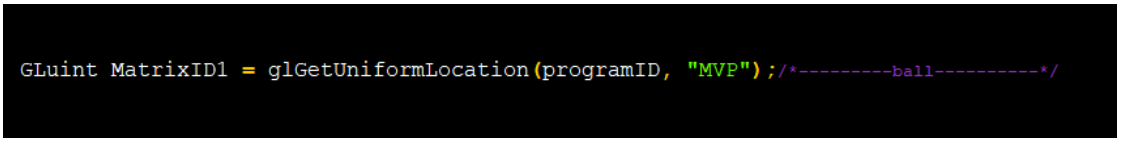
- Ερώτημα iv

Με αντίστοιχο τρόπο όπως στο ερώτημα i δημιουργούμε το ball1.obj το οποίο αναπαριστά μία σφαίρα φωτιάς με διάμετρο 15m και με υφή fire.jpg. Όταν ο χρήστης πατάει το πλήκτρο B η μπάλα εμφανίζεται σε ένα τυχαίο σημείο στο διάστημα xε(0,200), yε(0,200) και z=20. Στο κώδικα μας, όπως φαίνεται στις παρακάτω εικόνες έχουμε δημιουργήσει ένα MVP πίνακα, έχουμε διαβάσει το texture fire.jpg , ball1.obj και το έχουμε φορτώσει σε ένα VBO. Τέλος, στην εικόνα 6 φαίνεται αναλυτικά στο κώδικα ο MVP πίνακας που δημιουργήσαμε καθώς επίσης και ο τρόπος που εμφανίζονται τα τυχαία σημεία στο διάστημα που αναφέραμε.



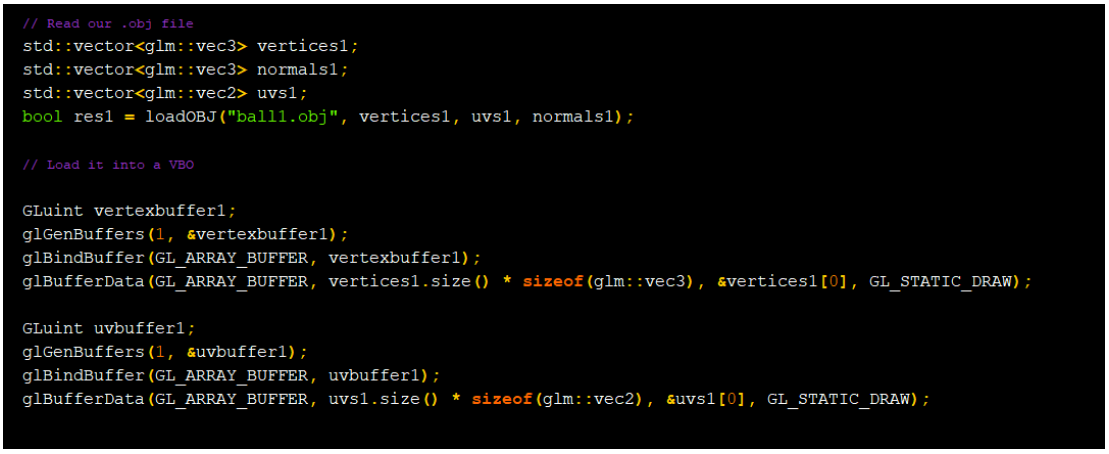
```
/************************ball*****************************************/
GLuint VertexArrayID1;
glGenVertexArrays(1, &VertexArrayID1);
glBindVertexArray(VertexArrayID1);
/************************boom*****************************************/
```

Εικόνα 3



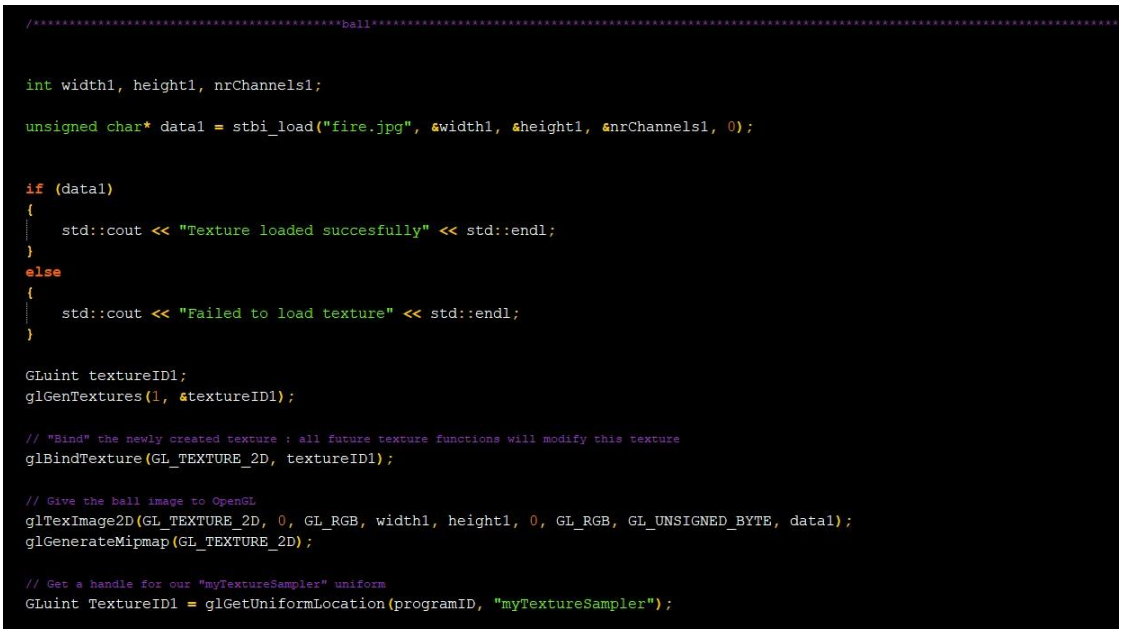
```
GLuint MatrixID1 = glGetUniformLocation(programID, "MVP");/*---------ball----------*/
```

Εικόνα 4



```
// Read our .obj file
std::vector<glm::vec3> vertices1;
std::vector<glm::vec3> normals1;
std::vector<glm::vec2> uvs1;
bool res1 = loadOBJ("ball1.obj", vertices1, uvs1, normals1);

// Load it into a VBO

GLuint vertexbuffer1;
glGenBuffers(1, &vertexbuffer1);
glBindBuffer(GL_ARRAY_BUFFER, vertexbuffer1);
glBufferData(GL_ARRAY_BUFFER, vertices1.size() * sizeof(glm::vec3), &vertices1[0], GL_STATIC_DRAW);

GLuint uvbuffer1;
glGenBuffers(1, &uvbuffer1);
glBindBuffer(GL_ARRAY_BUFFER, uvbuffer1);
glBufferData(GL_ARRAY_BUFFER, uvs1.size() * sizeof(glm::vec2), &uvs1[0], GL_STATIC_DRAW);
```

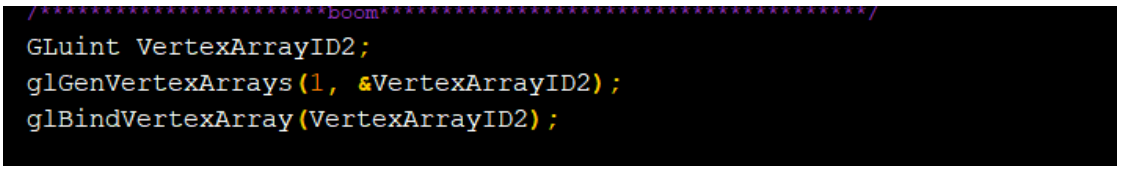
Εικόνα 5



```
/*****************************************ball*****************************************************************************************

int width1, height1, nrChannels1;

unsigned char* data1 = stbi_load("fire.jpg", &width1, &height1, &nrChannels1, 0);


if (data1)
{
    std::cout << "Texture loaded succesfully" << std::endl;
}
else
{
    std::cout << "Failed to load texture" << std::endl;
}

GLuint textureID1;
glGenTextures(1, &textureID1);

// "Bind" the newly created texture : all future texture functions will modify this texture
glBindTexture(GL_TEXTURE_2D, textureID1);

// Give the ball image to OpenGL
glTexImage2D(GL_TEXTURE_2D, 0, GL_RGB, width1, height1, 0, GL_RGB, GL_UNSIGNED_BYTE, data1);
glGenerateMipmap(GL_TEXTURE_2D);

// Get a handle for our "myTextureSampler" uniform
GLuint TextureID1 = glGetUniformLocation(programID, "myTextureSampler");
```

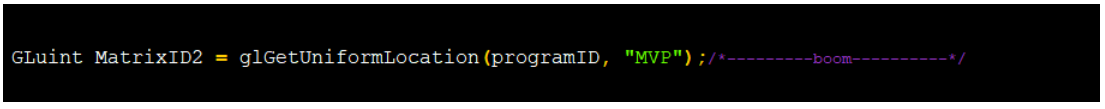
Εικόνα 6

Ολοκληρώνοντας, με αντίστοιχο τρόπο φτιάξαμε ένα boom.obj το οποίο θα φορτώνεται όταν η μπάλα προσγειώνεται πάνω σε ένα συγκεκριμένο σημείο του πλέγματος, ο κώδικας παρατίθεται παρακάτω.



```
/*********************boom*****************************************/
GLuint VertexArrayID2;
glGenVertexArrays(1, &VertexArrayID2);
glBindVertexArray(VertexArrayID2);
```

Εικόνα 7



```
GLuint MatrixID2 = glGetUniformLocation(programID, "MVP");/*---------boom----------*/
```

Εικόνα 8

```
int width2, height2, nrChannels2;

unsigned char* data2 = stbi_load("crater2.jpg", &width2, &height2, &nrChannels2, 0);


if (data2)
{
    std::cout << "Texture loaded succesfully" << std::endl;
}
else
{
    std::cout << "Failed to load texture" << std::endl;
}

GLuint textureID2;
glGenTextures(1, &textureID2);

// "Bind" the newly created texture : all future texture functions will modify this texture
glBindTexture(GL_TEXTURE_2D, textureID2);

// Give the boom image to OpenGL
glTexImage2D(GL_TEXTURE_2D, 0, GL_RGB, width2, height2, 0, GL_RGB, GL_UNSIGNED_BYTE, data2);
glGenerateMipmap(GL_TEXTURE_2D);

// Get a handle for our "myTextureSampler" uniform
GLuint TextureID2 = glGetUniformLocation(programID, "myTextureSampler");

// Read our .obj file
std::vector<glm::vec3> vertices2;
std::vector<glm::vec3> normals2;
std::vector<glm::vec2> uvs2;
bool res2 = loadOBJ("boom.obj", vertices2, uvs2, normals2);
```

Εικόνα 9

```
// Read our .obj file
std::vector<glm::vec3> vertices2;
std::vector<glm::vec3> normals2;
std::vector<glm::vec2> uvs2;
bool res2 = loadOBJ("boom.obj", vertices2, uvs2, normals2);

// Load it into a VBO
GLuint vertexbuffer2;
glGenBuffers(1, &vertexbuffer2);
glBindBuffer(GL_ARRAY_BUFFER, vertexbuffer2);
glBufferData(GL_ARRAY_BUFFER, vertices2.size() * sizeof(glm::vec3), &vertices2[0], GL_STATIC_DRAW);

GLuint uvbuffer2;
glGenBuffers(1, &uvbuffer2);
glBindBuffer(GL_ARRAY_BUFFER, uvbuffer2);
glBufferData(GL_ARRAY_BUFFER, uvs2.size() * sizeof(glm::vec2), &uvs2[0], GL_STATIC_DRAW);
```

Εικόνα 10

```
/*********************************************boom*************************************************************************/


    /*camera_function();
    glm::mat4 ProjectionMatrix2 = getProjectionMatrix();
    glm::mat4 ViewMatrix2 = getViewMatrix();
    glm::mat4 ModelMatrix2 = glm::mat4(1.0);
    glm::mat4 MVP2 = ProjectionMatrix2 * ViewMatrix2 * ModelMatrix2;


    glUniformMatrix4fv(MatrixID2, 1, GL_FALSE, &MVP2[0][0]);
    glBindTexture(GL_TEXTURE_2D, textureID2);
    glUniform1i(TextureID2, 0);


    // 1rst attribute buffer : vertices
    glEnableVertexAttribArray(0);
    glBindBuffer(GL_ARRAY_BUFFER, vertexbuffer2);
    glVertexAttribPointer(
        0,                  // attribute
        3,                  // size
        GL_FLOAT,           // type
        GL_FALSE,           // normalized?
        0,                  // stride
        (void*)0            // array buffer offset
    );
```

Εικόνα 11

```
// 2nd attribute buffer : UVs
glEnableVertexAttribArray(1);
glBindBuffer(GL_ARRAY_BUFFER, uvbuffer2);
glVertexAttribPointer(
    1,                                // attribute
    2,                                // size
    GL_FLOAT,                         // type
    GL_FALSE,                         // normalized?
    0,                                // stride
    (void*)0                          // array buffer offset
);



// Draw the triangle !
glDrawArrays(GL_TRIANGLES, 0, vertices2.size());
glDisableVertexAttribArray(0);
glDisableVertexAttribArray(1);


*/
```

Εικόνα 12

- Περιγραφή δυσκολιών

Τα σκέλη που δεν καταφέραμε να υλοποιήσουμε είναι στο ερώτημα α τους ελληνικούς χαρακτήρες στον τίτλο του παραθύρου στο ερώτημα iv αρχικά το fireball κάνει rendering στην αρχή των αξόνων πριν πατηθεί το πλήκτρο B η εικόνα αρχικοποιείται στο (0,0,0) με αποτέλεσμα να μην εκτελείται εξ ολοκλήρου η κίνηση που μας ζητείται. Κλείνοντας, η εμφάνιση του boom.obj που αφορά τον κρατήρα έχει μπει σε σχόλια (αν βγάλετε τα σχόλια μπορείτε να δείτε τον κρατήρα στην οθόνη) διότι δεν καταφέραμε να το υλοποιήσουμε σωστά.

ΠΛΗΡΟΦΟΡΙΕΣ ΣΧΕΤΙΚΑ ΜΕ ΤΗΝ ΥΛΟΠΟΙΗΣΗ

Το λειτουργικό σύστημα είναι windows και το περιβάλλον που υλοποιήσαμε την εργασία είναι Visual Studio x64.

ΑΞΙΟΛΟΓΗΣΗ ΛΕΙΤΟΥΡΓΙΑΣ ΤΗΣ ΟΜΑΔΑΣ

Η λειτουργία της ομάδας ήταν αποτελεσματική καθώς οργανώσαμε τα μέρη της άσκησης, έψαξε και δούλεψε ο καθένας ξεχωριστά και στην συνέχεια μαζί λύσαμε τις απορίες μας κα ολοκληρώσαμε την άσκηση.

ΑΝΑΦΟΡΕΣ

- Οι πηγές που αξιοποιήθηκαν κατά την εκπόνηση της άσκησης ήταν τα βιντεάκια από το εργαστήριο της Κυρίας Σταμάτη.
- https://learnopengl.com/Getting-started/Camera?fbclid=IwAR0BKLgs4QhDLS1CXE-zdD1IMUYd3AfggECxdHnluZesIqBlDSM4UXO4qx4
- http://www.opengl-tutorial.org/