ΠΡΟΓΡΑΜΜΑΤΙΣΤΙΚΗ ΑΣΚΗΣΗ 1-Β

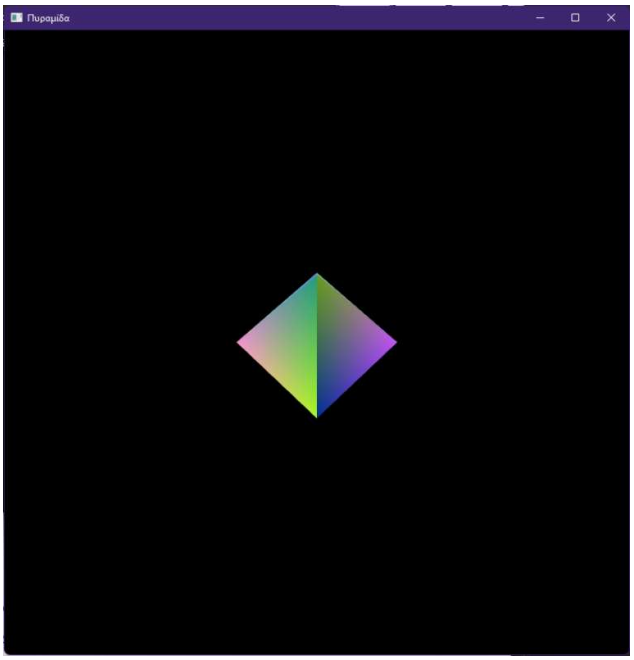
Γραφικά Υπολογιστών και Συστήματα Αλληλεπίδρασης - ΜΥΥ702

Ακαδ. Έτος 2021-2022

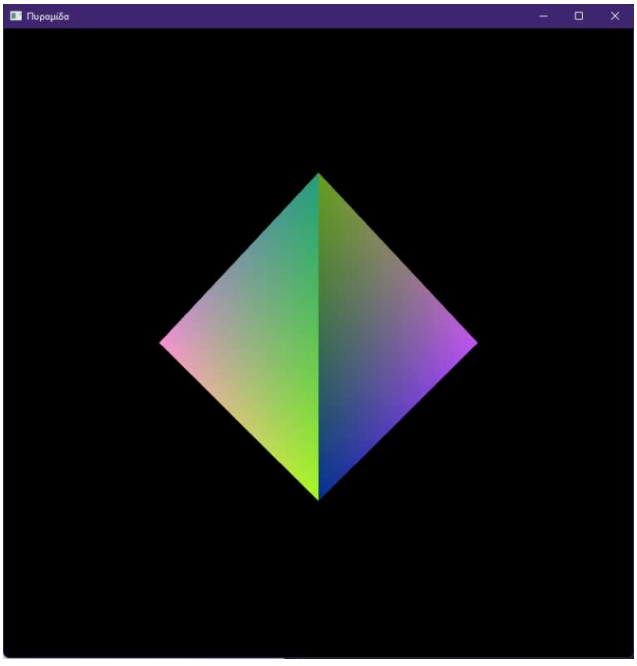
ΚΙΚΙΔΗΣ ΚΩΝΣΤΑΝΤΙΝΟΣ (4387)
ΤΣΑΜΠΙΡΑΣ ΚΩΝΣΤΑΝΤΙΝΟΣ (4508)

Λειτουργία του προγράμματος

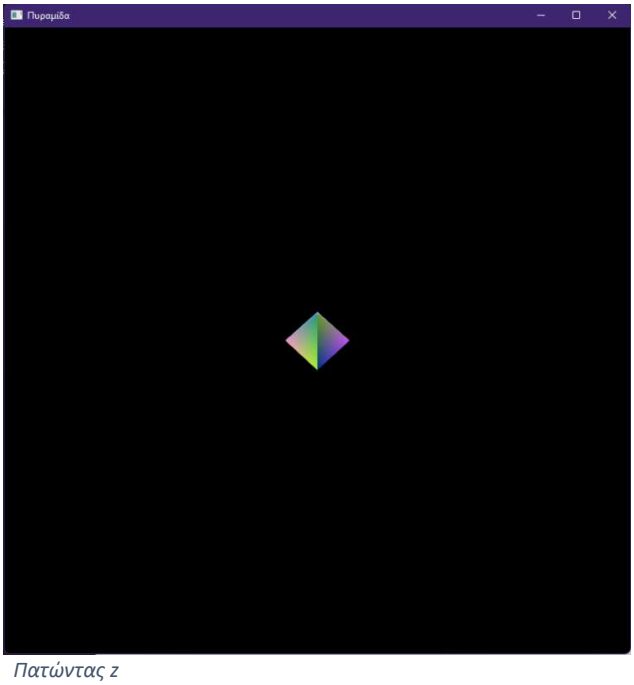
Το πρόγραμμα, αρχικά σχηματίζει μία πυραμίδα, αποτελούμενη από 1 τετράγωνο ως βάση (σχηματισμένο από 2 τρίγωνα) και 4 ισοσκελή και ίσα τρίγωνα, όπου κάθε σημείο της πυραμίδας έχει ένα τυχαία παραγμένο χρώμα και η κορυφή της πυραμίδας βρίσκεται πάνω στον άξονα z σε τυχαίο ύψος H που λαμβάνει τιμές στο διάστημα [2.0-10.0].



Όταν ο χρήστης πατά τα πλήκτρα **q** και **z** (πεζά γράμματα), τότε γίνεται κλιμάκωση της πυραμίδας, μεγαλώνοντας ή μικραίνοντας την αντίστοιχα.



Πατώντας q



Πατώντας z

Προβλήματα που υπάρχουν

Δεν έχουμε καταφέρει να υλοποιήσουμε την κίνηση της κάμερας στους άξονες που ζητά η εκφώνηση, είχαμε χρησιμοποιήσει σαν πρότυπο το controls.cpp χωρίς ουσιαστικό αποτέλεσμα.

Ιδιαιτερότητες/Ειδικές συνθήκες

Η πυραμίδα δεν κάνει scale αν πατηθεί το q ή το z ενώ είναι πατημένα τα Caps Lock και Shift μαζί, μόνο όταν είναι σκέτα πατημένα, χωρίς Caps Lock και χωρίς Shift.

Ανάλυση του κώδικα που χρησιμοποιήθηκε

Σαν βάση χρησιμοποιήθηκε ο κώδικας από το 1η εργασία σαν βάση (Source.cpp) όπου έχουν γίνει οι κατάλληλες τροποποιήσεις για να ανταποκρίνεται στις απαιτήσεις της τρέχουσας εργασίας και το controls.cpp για τον έλεγχο της κάμερας.

Στο Source.cpp

Έχουν συμπεριληφθεί οι εξής βιβλιοθήκες στο αρχείο στις γραμμές 15, 26, 27, 32.

```
#include <Windows.h>
#include <glm/gtc/matrix_transform.hpp>
#include <glm/gtx/transform.hpp>
#include "controls.hpp"
```

Αλλαγή Τίτλου Παραθύρου στη γραμμή 147

```
window = glfwCreateWindow(800, 800, "Πυραμίδα", NULL, NULL);
```

Αλλαγή background χρώματος σε μαύρο και ενεργοποίηση του βάθους για τον σωστό σχηματισμό των τρισδιάστατων σχημάτων (να φαίνονται αυτά που είναι πιο κοντά σε εμάς, και να κρύβονται αυτά που είναι πιο μακριά) στις γραμμές 168-174

```
// Black background
glClearColor(0.0f, 0.0f, 0.0f, 0.0f);
// Enable depth test
glEnable(GL_DEPTH_TEST);
// Accept fragment if it closer to the camera than the former one
glDepthFunc(GL_LESS);
```

Στη γραμμή 181 έχουμε βάλει να φορτώνει τους κατάλληλους shaders

```
GLuint programID = LoadShaders("TransformVertexShader.vertexshader",
"ColorFragmentShader.fragmentshader");
```

Στις γραμμές 186-197, τοποθετούμε την κάμερα στην θέση 10,10,30, και φτιάχνουμε την MVP matrix

```
GLuint MatrixID = glGetUniformLocation(programID, "MVP");
glm::mat4 Projection = glm::perspective(glm::radians(30.0f), 1.0f / 1.0f, 0.1f,
100.0f);
glm::mat4 View = glm::lookAt(
      glm::vec3(10, 10, 30),
      glm::vec3(0, 0, 0),
      glm::vec3(0, 0, 1)
);
glm::mat4 Model = glm::mat4(1.0f);
glm::mat4 MVP = Projection * View * Model;
```

Στις γραμμές 199-200, παράγουμε την τυχαία τιμή του ύψους H που θα βρίσκεται η κορυφή της πυραμίδας.

```
srand((unsigned)time(NULL));
float H = 2.0 + (((float)rand() / RAND_MAX) * (10.0 - 2.0));
```

Στις γραμμές 202-246 ορίζουμε τις θέσεις των σημείων των τριγώνων που σχηματίζουν την πυραμίδα και τα χρώματα των κορυφών (τα χρώματα παράχθηκαν τυχαία).

```
static const GLfloat g_vertex_buffer_data[] = {…};
static const GLfloat g_color_buffer_data[] = {…};
```

Στις γραμμές 254-260, έχουμε προσθέσει τον colorBuffer και τυπώνουμε την τιμή του H για να βλέπουμε πως σε κάθε εκτέλεση, πράγματι, αλλάζει το ύψος.

```
GLuint colorbuffer;
glGenBuffers(1, &colorbuffer);
glBindBuffer(GL_ARRAY_BUFFER, colorbuffer);
glBufferData(GL_ARRAY_BUFFER, sizeof(g_color_buffer_data), g_color_buffer_data,
GL_STATIC_DRAW);

printf("H = %f\n", H);
```

Στις γραμμές 263-290, υπολογίζουμε τους πίνακες με βάση την είσοδο του πληκτρολογίου για να μετακινούμε την κάμερα και μεγαλώνουμε ή μικραίνουμε την πυραμίδα μας πατώντας q ή z (πεζά) αντίστοιχα, αυτό γίνεται σταδιακά με βήμα 0.001.

```
glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
glUseProgram(programID);
computeMatricesFromInputs();
glm::mat4 ProjectionMatrix = getProjectionMatrix();
glm::mat4 ViewMatrix = getViewMatrix();
glm::mat4 ModelMatrix = Model;
if ((GetKeyState('Q') & 0x8000) && (!((GetKeyState(VK_SHIFT) & 0x8000) ^
(GetKeyState(VK_CAPITAL) & 1)))) {
      printf("DEBUG: q pressed\n");
      Model = glm::scale(ModelMatrix, glm::vec3(1.001f, 1.001f, 1.001f));
}

if ((GetKeyState('Z') & 0x8000) && (!((GetKeyState(VK_SHIFT) & 0x8000) ^
(GetKeyState(VK_CAPITAL) & 1)))) {
      printf("DEBUG: z pressed\n");
      Model = glm::scale(ModelMatrix, glm::vec3(0.999f, 0.999f, 0.999f));
}
glm::mat4 MVP = ProjectionMatrix * ViewMatrix * ModelMatrix;
glUniformMatrix4fv(MatrixID, 1, GL_FALSE, &MVP[0][0]);
```

Στις γραμμές 304-314 βάζουμε τα χρώματα

```
// 2nd attribute buffer : colors
glEnableVertexAttribArray(1);
glBindBuffer(GL_ARRAY_BUFFER, colorbuffer);
glVertexAttribPointer(
      1,
      3,
      GL_FLOAT,
      GL_FALSE,
      0,
      (void*)0
);
```

Και στις γραμμές 328-329, ελέγχουμε αν έχει πατηθεί το Q (κεφαλαίο) για να τερματίσουμε το πρόγραμμα (ή αν ο χρήστης πατήσει το X του παραθύρου για τερματισμό). Για να ανιχνεύσουμε αν το Q είναι πεζό ή κεφαλαίο χρησιμοποιούμε την βιβλιοθήκη των Windows και την συνάρτηση GetKeyState().

```
while ((!(GetKeyState('Q') & 0x8000) || (!(GetKeyState(VK_SHIFT) & 0x8000) ^
(GetKeyState(VK_CAPITAL) & 1))) && glfwWindowShouldClose(window) == 0);
```

Στο controls.cpp

Στη γραμμή 24, αλλάξαμε την αρχική θέση της κάμερας σε (10,10,30)

```
glm::vec3 position = glm::vec3( 10, 10, 30 );
```

Απενεργοποιήσαμε τις γραμμές 46-65, που αφορούν την κίνηση της κάμερας με το ποντίκι

Στην γραμμή 66 ορίσαμε την κάμερα να κοιτά με κατεύθυνση προς το κέντρο

```
glm::vec3 direction(-10,-10,-30);
```

Στις γραμμές 109-115, πειράξαμε το projectionMatrix και το viewMatrix, ώστε το aspect ratio να είναι 1:1 και η lookAt να κοιτά στο κέντρο και ο κατακόρυφος άξονας να είναι ο z.

```
ProjectionMatrix = glm::perspective(glm::radians(FoV), 1.0f / 1.0f, 0.1f, 100.0f);
ViewMatrix       = glm::lookAt(
                        position,
                        direction,//glm::vec3(0, 0, 0),//position+direction
                        glm::vec3(0, 0, 1)//up
                    );
```