Γραφικά Υπολογιστων και συστήματα αλληλεπίδρασης

Προγραμματιστική άσκηση 1Γ

Νεφέλη-Ελένη Κατσιλέρου 4385

Χρήστος Καραγιαννίδης 4375

Σχολια:

1) Καναμε include την <windows.h> για τα κεφαλαια.
2) Γραψαμε τον κωδικα σε visual studio σε windows.
3) Στην αρχη ειχαμε ορισμενα errors για λογους που δεν καταλαβαιναμε αλλα λυθηκαν με το να κανουμε δεξι κλικ στο project στο solution explorer > add > new item και προσθετωντας ολα τα αρχεια του φακελου που βρισκεται το project συμπεριλαμβανομενου και των αρχειων .obj και .jpg.
4) Τα 3 αρχεια .obj για να μην προκαλουν error κανουμε δεξι κλικ πανω τους >properties > Exclude from build και βαζουμε ως επιλογη «Yes».
5) Παρατηρησαμε οτι οι ταχυτητες περιστροφης πλανητη, κινησης μετεοριτη, zoom in/out , καμερας ειναι διαφορετικες για τον ιδιο κωδικα σε διαφορετικους υπολογιστες.
6) Για να βλεπουμε καλυτερα τους πλανητες βαλαμε την αρχικη θεση της καμερας στο 10,10,80 με upvector (0,1,0)
7) Η main μας ονομαζεται My_test.cpp

i)

Στο πρωτο ερωτημα χρησιμοποιωντας τον κωδικα που ειχαμε απο την ασκηση 1Β καναμε την αλλαγη για το βασικο παραθυρο σε 800x800 καθως επισης αλλαξαμε , το χρωμα του background και αλλαξαμε το πληκτρο τερματισμου σε <Q> (κεφαλαιο).

```
270     window = glfwCreateWindow(800, 800, u8"Ηλιακο Συστημα", NULL, NULL);
```

```
191     glClearColor(0.0f, 0.0f, 0.0f, 0.0f);
```

Αφου ζητηθηκε το κλειδί τερματισμού να ειναι μονο το uppercase **Q** καναμε τις εξής αλλαγές (οι οποιες ειναι ιδιες με την ασκηση 1Β) :

1)

```
128 bool CheckKeyState() {
129     if ((GetKeyState(VK_CAPITAL) & 0X0001) == 1) {
130         return true;
131     }
132
133     return false;
134 }
```

Φτιάξαμε την συνάρτηση CheckKeyState() η οποία επιστρεφει true (1) όταν το CAPS_LOCK ειναι on και αντίθετα false (0).

2)

```
132 int main(void)
133 {
134     // Initialise GLFW
135     int termination=0;
```

Αρχικοποιήσαμε την μεταβλητη termination την οποία χρησιμοποιούμε παρακάτω.

3)

```
338             if (CheckKeyState()==true) {
339                 termination =1;
340             }
341             else if (CheckKeyState()==false){
342                 termination = 0;
343             }
```

Λιγες γραμμές πριν το τέλος της do.. while() ελεγχουμε αν η CheckKeyState() μας επιστρέφει true (το οποίο σημαίνει οτι αν δεχτούμε το key_Q θα ειναι uppercase γιατι το CAPS LOCK ειναι on). Σε αυτη την περίπτωση το termination γινεται ισο με 1.

4)

```
348     while (glfwGetKey(window, GLFW_KEY_Q) != GLFW_PRESS || termination == 0
349         && glfwWindowShouldClose(window) == 0);
```

Στην while προσθέσαμε μια extra παραμετρο η οποια ελεγχει αν το CAPS LOCK ειναι on οπου σε αυτή την περιπτωση αν δεχτεί και το κλειδί **Q** τερματιζει.

ii) **My_test.cpp**

1) Καναμε include το objloader.hpp
2) Προσθεσαμε την συναρτηση loadOBJ η οποια βρισκεται στα αρχεια που δωθηκαν στο obj_and_texture_example φακελο.

3) Για καθε object που θα χρειαστουμε (3) φτιαξαμε ενα vertexArrayID

```
299     GLuint VertexArrayID;
300     glGenVertexArrays(1, &VertexArrayID);
301     glBindVertexArray(VertexArrayID);
302
303     GLuint VertexArrayID2;
304     glGenVertexArrays(1, &VertexArrayID2);
305     glBindVertexArray(VertexArrayID2);
306
307     GLuint VertexArrayID3;
308     glGenVertexArrays(1, &VertexArrayID3);
309     glBindVertexArray(VertexArrayID3);
```

4) Αλλαξαμε τους shaders που φορτωνουμε ωστε να ειναι οι καινουριοι TransformVertexShader και TextureFragmentShader
5) Αρχικοποιησαμε εναν πινακα 3 θεσεων τον οποιον τον ονομασαμε textureTEST και επειτα δηλωνουμε οτι στις θεσεις του textureTEST θα μπουν 3 διαφορετικα textures. Μετα για καθε ενα texture που θελουμε να φορτωσουμε κανουμε bind το texture, κανουμε load την αντιστοιχη εικονα στο dataTEST και τελος κανουμε free την dataTEST ετσι ωστε να

μπορουμε να την ξαναχρησιμοποιησουμε για το επομενο texture.

```
319     int width, height, nrChannels;
320
321
322     GLuint textureTEST[3];
323     glGenTextures(3, textureTEST);
324
325     glBindTexture(GL_TEXTURE_2D, textureTEST[0]);
326     unsigned char* dataTEST = stbi_load("sun.jpg", &width, &height, &nrChannels, 0);
327     glTexImage2D(GL_TEXTURE_2D, 0, GL_RGB, width, height, 0, GL_RGB, GL_UNSIGNED_BYTE, dataTEST);
328     glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST);
329     glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
330     stbi_image_free(dataTEST);
331
332     glBindTexture(GL_TEXTURE_2D, textureTEST[1]);
333     dataTEST = stbi_load("planet.jpg", &width, &height, &nrChannels, 0);
334     glTexImage2D(GL_TEXTURE_2D, 0, GL_RGB, width, height, 0, GL_RGB, GL_UNSIGNED_BYTE, dataTEST);
335     glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST);
336     glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
337     stbi_image_free(dataTEST);
338
339     glBindTexture(GL_TEXTURE_2D, textureTEST[2]);
340     dataTEST = stbi_load("meteor.jpg", &width, &height, &nrChannels, 0);
341     glTexImage2D(GL_TEXTURE_2D, 0, GL_RGB, width, height, 0, GL_RGB, GL_UNSIGNED_BYTE, dataTEST);
342     glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST);
343     glTexParameterf(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
344     stbi_image_free(dataTEST);
345
```

6) Αντιγραψαμε απο το tutorial07.cpp τον κωδικα με τον οποιο διαβαζουμε το obj αρχειο και φτιαξαμε αλλους 2 για τα 2 επιπλεον objects.

```
357     // Read our .obj file
358     std::vector<glm::vec3> vertices;
359     std::vector<glm::vec3> normals;
360     std::vector<glm::vec2> uvs;
361     bool res = loadOBJ("sun.obj", vertices, uvs, normals);
362
363     std::vector<glm::vec3> vertices2;
364     std::vector<glm::vec3> normals2;
365     std::vector<glm::vec2> uvs2;
366     bool res2 = loadOBJ("planet.obj", vertices2, uvs2, normals2);
367
368     std::vector<glm::vec3> vertices3;
369     std::vector<glm::vec3> normals3;
370     std::vector<glm::vec2> uvs3;
371     bool res3 = loadOBJ("meteor.obj", vertices3, uvs3, normals3);
```

7) Φτιαξαμε εναν vertexbuffer και εναν uvbuffer για καθε object.

```
376     GLuint vertexbuffer;
377     glGenBuffers(1, &vertexbuffer);
378     glBindBuffer(GL_ARRAY_BUFFER, vertexbuffer);
379     glBufferData(GL_ARRAY_BUFFER, vertices.size() * sizeof(glm::vec3), &vertices[0], GL_STATIC_DRAW);
380
381     GLuint uvbuffer;
382     glGenBuffers(1, &uvbuffer);
383     glBindBuffer(GL_ARRAY_BUFFER, uvbuffer);
384     glBufferData(GL_ARRAY_BUFFER, uvs.size() * sizeof(glm::vec2), &uvs[0], GL_STATIC_DRAW);
385
386
387
388
389     GLuint vertexbuffer2;
390     glGenBuffers(1, &vertexbuffer2);
391     glBindBuffer(GL_ARRAY_BUFFER, vertexbuffer2);
392     glBufferData(GL_ARRAY_BUFFER, vertices2.size() * sizeof(glm::vec3), &vertices2[0], GL_STATIC_DRAW);
393
394     GLuint uvbuffer2;
395     glGenBuffers(1, &uvbuffer2);
396     glBindBuffer(GL_ARRAY_BUFFER, uvbuffer2);
397     glBufferData(GL_ARRAY_BUFFER, uvs2.size() * sizeof(glm::vec2), &uvs2[0], GL_STATIC_DRAW);
398
399     GLuint vertexbuffer3;
400     glGenBuffers(1, &vertexbuffer3);
401     glBindBuffer(GL_ARRAY_BUFFER, vertexbuffer3);
402     glBufferData(GL_ARRAY_BUFFER, vertices3.size() * sizeof(glm::vec3), &vertices3[0], GL_STATIC_DRAW);
403
404     GLuint uvbuffer3;
405     glGenBuffers(1, &uvbuffer3);
406     glBindBuffer(GL_ARRAY_BUFFER, uvbuffer3);
407     glBufferData(GL_ARRAY_BUFFER, uvs3.size() * sizeof(glm::vec2), &uvs3[0], GL_STATIC_DRAW);
408
```

8) Μεσα στην do..while φτιαξαμε 2 ακομα model Matrices και 2 ακομα MVP (εναν για καθε object).Για καθε object που θα θελησουμε να αλλαξουμε το model Matrix του καλουμε μια συναρτηση getModelMatrix() που βρισκεται στο **controls.cpp** ωστε να μπορουμε να κανουμε μετασχηματισμους σε αυτον τον πινακα εκει.

```
421         computeMatricesFromInputs();
422         glm::mat4 ProjectionMatrix = getProjectionMatrix();
423         glm::mat4 ViewMatrix = getViewMatrix();
424         glm::mat4 ModelMatrix = glm::mat4(1.0);
425         glm::mat4 ModelMatrix2 = getModelMatrix2();
426         glm::mat4 ModelMatrix3 = getModelMatrix3();
427         glm::mat4 MVP = ProjectionMatrix * ViewMatrix * ModelMatrix;
428         glm::mat4 MVP2 = ProjectionMatrix * ViewMatrix * ModelMatrix2;
429         glm::mat4 MVP3 = ProjectionMatrix * ViewMatrix * ModelMatrix3;
430
```

9) Για καθε ενα απο τα objects μας καλουμε την glUniformMatrix4fv για τον αντιστοιχο MVP πινακα κανουμε activeTexture τον πινακα με τα 3 textures μας και κανουμε bind το σωστο texture. Για τα 2 attribute buffers που εχουμε εναν για τα vertices και ενα για τα uvs κανουμε το καταλληλο enable bindBuffer και επειτα ζωγραφιζουμε το σχημα που εχει το texture πανω του.

```
433         glUniformMatrix4fv(MatrixID, 1, GL_FALSE, &MVP[0][0]);
434
435
436         // Bind our texture in Texture Unit 0
437         glActiveTexture(GL_TEXTURE0);
438         glBindTexture(GL_TEXTURE_2D, textureTEST[0]);//                                  SUN
439         // Set our "myTextureSampler" sampler to use Texture Unit 0
440         glUniform1i(*textureTEST, 0);
441
442         // 1rst attribute buffer : vertices
443         glEnableVertexAttribArray(0);
444         glBindBuffer(GL_ARRAY_BUFFER, vertexbuffer);
445         glVertexAttribPointer(
446             0,                  // attribute
447             3,                  // size
448             GL_FLOAT,           // type
449             GL_FALSE,           // normalized?
450             0,                  // stride
451             (void*)0            // array buffer offset
452         );
453
454         // 2nd attribute buffer : UVs
455         glEnableVertexAttribArray(1);
456         glBindBuffer(GL_ARRAY_BUFFER, uvbuffer);
457         glVertexAttribPointer(
458             1,                                // attribute
459             2,                                // size
460             GL_FLOAT,                         // type
461             GL_FALSE,                         // normalized?
462             0,                                // stride
463             (void*)0                          // array buffer offset
464         );
465
466         // Draw the triangle !
467         glDrawArrays(GL_TRIANGLES, 0, vertices.size());
```

```
469         glUniformMatrix4fv(MatrixID, 1, GL_FALSE, &MVP2[0][0]);
470
471         glActiveTexture(GL_TEXTURE0);
472         glBindTexture(GL_TEXTURE_2D, textureTEST[1]);//                                  PLANET
473         glUniform1i(*textureTEST, 0);
474
475         // 1rst attribute buffer : vertices
476         glEnableVertexAttribArray(0);
477         glBindBuffer(GL_ARRAY_BUFFER, vertexbuffer2);
478         glVertexAttribPointer(
479             0,                  // attribute
480             3,                  // size
481             GL_FLOAT,           // type
482             GL_FALSE,           // normalized?
483             0,                  // stride
484             (void*)0            // array buffer offset
485         );
486
487         // 2nd attribute buffer : UVs
488         glEnableVertexAttribArray(1);
489         glBindBuffer(GL_ARRAY_BUFFER, uvbuffer2);
490         glVertexAttribPointer(
491             1,                                // attribute
492             2,                                // size
493             GL_FLOAT,                         // type
494             GL_FALSE,                         // normalized?
495             0,                                // stride
496             (void*)0                          // array buffer offset
497         );
498
499         // Draw the triangle !
500         glDrawArrays(GL_TRIANGLES, 0, vertices2.size());
```

```
503         glUniformMatrix4fv(MatrixID, 1, GL_FALSE, &MVP3[0][0]);
504
505         glActiveTexture(GL_TEXTURE0);
506         glBindTexture(GL_TEXTURE_2D, textureTEST[2]);//                                  METEOR
507         glUniform1i(*textureTEST, 0);
508
509         // 1rst attribute buffer : vertices
510         glEnableVertexAttribArray(0);
511         glBindBuffer(GL_ARRAY_BUFFER, vertexbuffer3);
512         glVertexAttribPointer(
513             0,                  // attribute
514             3,                  // size
515             GL_FLOAT,           // type
516             GL_FALSE,           // normalized?
517             0,                  // stride
518             (void*)0            // array buffer offset
519         );
520
521         // 2nd attribute buffer : UVs
522         glEnableVertexAttribArray(1);
523         glBindBuffer(GL_ARRAY_BUFFER, uvbuffer3);
524         glVertexAttribPointer(
525             1,                                // attribute
526             2,                                // size
527             GL_FLOAT,                         // type
528             GL_FALSE,                         // normalized?
529             0,                                // stride
530             (void*)0                          // array buffer offset
531         );
532
533         // Draw the triangle !
534         glDrawArrays(GL_TRIANGLES, 0, vertices3.size());
```

10) Για την σφαιρα S ανοιξαμε το αρχειο sun.obj στο visual studio και παρατηρησαμε οτι η ακτινα ηταν ηδη 15 και αφου το κεντρο των αντικειμενων ειναι ηδη το 0,0,0 δεν χρειαστηκε καποια μετατοπιση .

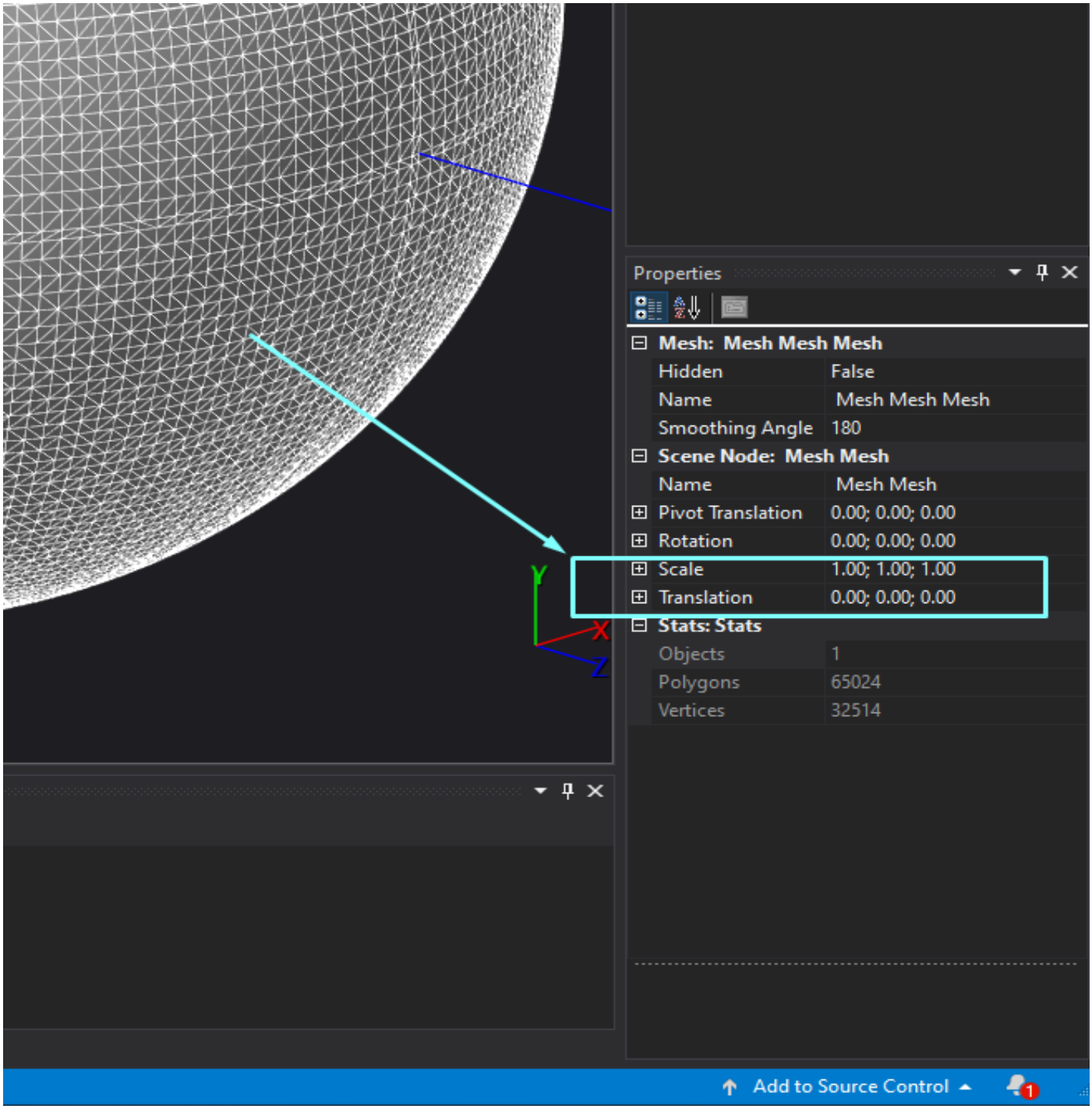
Για την σφαιρα P παλι δεν χρειαστηκε να αλλαξουμε την ακτινα γιατι ηταν ηδη 5 αλλα για να κανουμε την μετατοπιση της καναμε κλικ στην σφαιρα (το μοντελο) και στα properties κατω δεξια της οθονης καναμε το translation 25,0,0 ετσι ωστε καθε φορα το object να εμφανιζεται σε εκεινες τις συντεταγμενες.

Για την σφαιρα M επισης δεν αλλαξαμε την ακτινα γιατι ηταν ηδη 2 και η αρχικη της θεση ειανι στο 0,0,0 (κρυμμενη απο μεσα στον ηλιο).

Αν χρειαζοταν να ειχαμε αλλαξει την ακτινα η και την μετατοπιση καποιας αλλης σφαιρας θα το καναμε με τον ιδιο τροπο αλλαζοντας στα properties της αντιστοιχης σφαιρας τις παραμετρους scale(x,y,z) και Translation(x,y,z). Πχ αν η S ηταν ακτινας 5 και θελαμε να την κανουμε 15 θα αλλαζαμε την scale(x,y,z) σε scale(3.0,3.0,3.0).

Γνωριζουμε επισης οτι μπορουσαμε να κανουμε αυτες τις 2 αλλαγες και μεσα στον κωδικα με την χρηση των glm::scale και glm::translate (οπως εχουμε κανει στην ασκηση 1B) αλλαζοντας τους αντιστοιχους MVP ομως ειχαμε κανει ηδη αυτη την υλοποιηση και αφου ρωτησαμε και μας ειπατε οτι παρολο που δεν

ειναι ο ενδεδειγμενος τροπος δεν ειναι και λαθος οποτε μειναμε με αυτον τον τροπο.



Controls.cpp

Ζητειται η καμερα να κινειται γυρω απο τους αξονες του παγκοσμιου συστηματος συντεταγμενων.Η κινηση της καμερας γυρω απο τον αξονα x γινεται με τα πληκτρα <w>,<x> (lowercase) . Οταν δεχομαστε ενα απο τα 2 πληκτρα οριζουμε εναν πινακα 4x4 που τον ονομαζουμε rotationMatrix ο οποιος παιρνει τις τιμες του απο την glm::rotate κανοντας rotate τον ModelMatrix κατα 0.0005f γυρω απο τον αξονα x (1,0,0) του παγκοσμιου συστηματος συντεταγμενων .Επειτα οριζουμε ενα διανυσμα myVec 4 διαστασεων το οποιο παιρνει ως x,y,z τα x,y,z της θεσης της καμερας (position) και βαζουμε μια τεταρτη συντεταγμενη για να μπορουμε να κανουμε τον πολλαπλασιασμο με τον πινακα 4x4. Επειτα κανουμε τον πολλαπλασιασμο και αποθηκευουμε το αποτελεσμα του σε ενα καινουριο διανυσμα MyNewVec και επειτα παιρνουμε τις 3 πρωτες συντεταγμενες του (x,y,z) και τις αναθετουμε στις x,y,z του position της καμερας ωστε να αλλαξει η θεση της. Αντιστοιχως για να πετυχουμε την κινηση προς την αντιθετη μερια το μονο που αλλαζουμε ειναι στην rotate την φορα περιστροφης σε (-1,0,0).

Η κινηση της καμερας γυρω απο τον αξονα y γινεται με τα πληκτρα <a>,<d> (lowercase) . Οταν δεχομαστε ενα απο τα 2 πληκτρα οριζουμε εναν πινακα 4x4 που τον ονομαζουμε rotationMatrix ο οποιος παιρνει τις τιμες του απο την glm::rotate κανοντας rotate τον ModelMatrix κατα 0.0005f γυρω απο τον αξονα y του παγκοσμιου συστηματος συντεταγμενων ο οποιος ταυτιζεται με τον αξονα z (0,0,1) της πυραμιδας.Επειτα οριζουμε ενα διανυσμα myVec 4 διαστασεων το οποιο παιρνει ως x,y,z τα x,y,z της θεσης της καμερας (position) και βαζουμε μια τεταρτη συντεταγμενη για να μπορουμε να κανουμε τον πολλαπλασιασμο με τον πινακα 4x4. Επειτα κανουμε τον πολλαπλασιασμο και αποθηκευουμε το αποτελεσμα του σε ενα

καινουριο διανυσμα MyNewVec και επειτα παιρνουμε τις 3 πρωτες συντεταγμενες του (x,y,z) και τις αναθετουμε στις x,y,z του position της καμερας ωστε να αλλαξει η θεση της. Αντιστοιχως για να πετυχουμε την κινηση προς την αντιθετη μερια το μονο που αλλαζουμε ειναι στην rotate την φορα περιστροφης σε (0,0,-1).

```
174     if ((glfwGetKey(window, GLFW_KEY_A) == GLFW_PRESS) && (caps_state == 1)) {
175         glm::mat4 rotationMatrix = glm::rotate(ModelMatrix, 0.0005f, glm::vec3(0, 0, 1));
176         glm::vec4 myVec = glm::vec4(position[0], position[1], position[2], 0.1f);
177         glm::vec4 myNewVec = rotationMatrix * myVec;
178         for (int i = 0; i <= 2; i++) {
179             position[i] = myNewVec[i];
180         }
181     }
182     if ((glfwGetKey(window, GLFW_KEY_D) == GLFW_PRESS) && (caps_state == 1)) {
183         glm::mat4 rotationMatrix = glm::rotate(ModelMatrix, 0.0005f, glm::vec3(0, 0, -1));
184         glm::vec4 myVec = glm::vec4(position[0], position[1], position[2], 0.1f);
185         glm::vec4 myNewVec = rotationMatrix * myVec;
186         for (int i = 0; i <= 2; i++) {
187             position[i] = myNewVec[i];
188         }
189 
190     }
191     if ((glfwGetKey(window, GLFW_KEY_X) == GLFW_PRESS) && (caps_state == 1)) {
192         glm::mat4 rotationMatrix = glm::rotate(ModelMatrix, 0.0005f, glm::vec3(1, 0, 0));
193         glm::vec4 myVec = glm::vec4(position[0], position[1], position[2], 0.1f);
194         glm::vec4 myNewVec = rotationMatrix * myVec;
195         for (int i = 0; i <= 2; i++) {
196             position[i] = myNewVec[i];
197         }
198 
199     }
200     if ((glfwGetKey(window, GLFW_KEY_W) == GLFW_PRESS) && (caps_state == 1)) {
201         glm::mat4 rotationMatrix = glm::rotate(ModelMatrix, 0.0005f, glm::vec3(-1, 0, 0));
202         glm::vec4 myVec = glm::vec4(position[0], position[1], position[2], 0.1f);
203         glm::vec4 myNewVec = rotationMatrix * myVec;
204         for (int i = 0; i <= 2; i++) {
205             position[i] = myNewVec[i];
206         }
207 
208     }
```

Η οποια ειναι η υλοποιηση μας απο την προηγουμενη ασκηση 1Β.

Οσον αφορα το zoom in και zoom out σκεφτηκαμε μια καλυτερη υλοποιηση απο αυτην της προηγουμενης ασκησης.Σε αυτη την υλοποιηση ελεγχουμε την θεση της καμερας καθε στιγμη για να δουμε αν οι τιμες της x,y,z ειναι >=0 και αντιστοιχως οπου ειναι μεγαλυτερες του μηδενος και κανουμε zoom in (πηγαινουμε πιο κοντα προς το 0,0,0) , τις μειωνουμε κατα 0.1 επι την απολυτη τιμη του x,y,z αντιστοιχως επι deltaTime.

```
115     if (glfwGetKey(window, GLFW_KEY_EQUAL) == GLFW_PRESS) {
116
117
118         if (position[0] >= 0) {
119             position[0] -= 0.1 * abs(position[0]) * deltaTime;
120         }
121         else {
122             position[0] += 0.1 * abs(position[0]) * deltaTime;
123         }
124
125         if (position[1] >= 0) {
126             position[1] -= 0.1 * abs(position[1]) * deltaTime;
127         }
128         else {
129             position[1] += 0.1 * abs(position[1]) * deltaTime;
130         }
131
132         if (position[2] >= 0) {
133             position[2] -= 0.1 * abs(position[2]) * deltaTime;
134         }
135         else {
136             position[2] += 0.1 * abs(position[2]) * deltaTime;
137         }
138     }
139     if (glfwGetKey(window, GLFW_KEY_MINUS) == GLFW_PRESS) {
140
141         if (position[0] <= 0) {
142             position[0] -= 0.1 * abs(position[0]) * deltaTime;
143         }
144         else {
145             position[0] += 0.1 * abs(position[0]) * deltaTime;
146         }
147         if (position[1] <= 0) {
148             position[1] -= 0.1 * abs(position[1]) * deltaTime;
149         }
150         else {
151             position[1] += 0.1 * abs(position[1]) * deltaTime;
152         }
153         if (position[2] <= 0) {
154             position[2] -= 0.1 * abs(position[2]) * deltaTime;
155         }
156         else {
157             position[2] += 0.1 * abs(position[2]) * deltaTime;
158         }
159
160     }
```

Στο αρχειο controls.cpp οπως ειπαμε παραπανω ορισαμε 2 συναρτησεις για να επιστρεφουμε τον ModelMatrix 2 και 3 στην main (My_test.cpp) .

Ορισαμε τις μεταβλητες:

```
int collision
    Ειναι 1 αν ειχαμε συγκρουση του Μετεωριτη με τον πλανητη αλλιως 0.
int MeteorMovement
    Οσο ειναι 1 κινειται ο Μετεωριτης.
int able_to_press_space
    Οσο ειναι 0 δεν παιζει ρολο αν ξαναπατησει ο χρηστης το space και ξαναγινεται 1 οταν τελειωσει η κινηση του μετεωριτη
float Meteor_x
float Meteor_y
float Meteor_z
    οι συντεταγμενες του μετεωριτη (που αρχικη τιμη εχουν την θεση της καμερας)
float Planet_x
float Planet_y
float Planet_z
    οι συντεταγμενες του πλανητη ενω περιστρεφεται γυρω απο τον ηλιο
float speedOfX = 0.1;
    κατα ποσο μειωνεται/αυξανεται καθε φορα η θεση x του μετεωριτη αναλογως με το που βρισκεται.
float ratioX2Y
    κατα ποσο μειωνεται/αυξανεται καθε φορα η θεση y του μετεωριτη η οποια εξαρταται απο τον x
float ratioX2Z
    κατα ποσο μειωνεται/αυξανεται καθε φορα η θεση y του μετεωριτη η οποια εξαρταται απο τον x
```

```
float distance_of_planets = 0.0;
```

η αποσταση του κεντρου του πλανητη απο το κεντρο του μετεωριτη.

Ο πλανητης κινειται γυρω απο τον ηλιο σε μια κυκλικη τροχια με κεντρο το 0,0,0 . Οσο collision == 0 (αρα δεν υπαρχει συγκρουση) κανουμε rotate τον ModelMatrix2 (του planet) κατα 0.0005f rad γυρω απο τον αξονα y (αρα y = σταθερο , x,z μεταβαλομενα)

```
198    if (collision == 0){
199        ModelMatrix2 = glm::rotate(ModelMatrix2, 0.0005f, glm::vec3(0, 1, 0));
200
201    }
```

Με το πληκτρο space ξεκιναει η κινηση του μετεωριτη απο τη θεση της καμερας προς το κεντρο του ηλιου (0,0,0) ακολουθωντας μια νοητη γραμμη.

Τη στιγμη που ο χρηστης πατησει το space (δεδομενου οτι μπορει λογω της μεταβλητης able_to_press_space) η μεταβλητη able_to_press_space γινεται 0 ωστε να μην υπαρχουν πολλαπλα registers του space εμποδιζοντας να ξαναπατηθει το space μεχρι να τελειωσει η κινηση του μετεωριτη (ειτε με συγκρουση ειτε φτανοντας στο κεντρο του κυκλου). Η μεταβλητη Meteor_movement γινεται 1 ξεκινώντας την κινηση του μετεωριτη. Κανουμε μια φορα translate το ModelMatrix του μετεωριτη απο το 0,0,0 στην θεση της καμερας και δινουμε τις τιμες x,y,z αυτες στις μεταβλητες Meteor_x , Meteor_y, Meteor_z. Τελος οριζουμε το βημα με το οποιο θα μειωνεται/ αυξανεται το x,y,z του μετεωριτη με τις speedOfX , ratioX2Y , ratioX2Z.

```
207     if ((glfwGetKey(window, GLFW_KEY_SPACE) == GLFW_PRESS) && able_to_press_space == 1) {
208         able_to_press_space = 0;
209         MeteorMovement = 1;
210         ModelMatrix3 = glm::translate(ModelMatrix3, glm::vec3(position[0],position[1], position[2]));
211          Meteor_x = position[0];
212          Meteor_y = position[1];
213          Meteor_z = position[2];
214          printf("x=%f", Meteor_x);
215          printf("y=%f", Meteor_y);
216          printf("z=%f", Meteor_z);
217          speedOfX = 0.0005;
218          ratioX2Y = position[1] / position[0];
219          ratioX2Z = position[2] / position[0];
220
221
222
223     }
```

Αφου πλεον το MeteorMovement ειναι 1 και το collision ειναι 0 , ελεγχουμε για καθε χρονικη στιγμη (καθε κινηση του μετεωριτη) αν εχει γινει συγκρουση με τον πλανητη χρησιμοποιωντας την checkForCollision την οποια καλουμε δινωντας σαν ορισματα τις συντεταγμενες του Meteor . Επειτα μετατοπιζουμε τον Μετεωριτη (modelMatrix3) κατα (x,y,z) = (- speedOfX, -speedOfX * ratioX2Y, -speedOfX * ratioX2Z) προς το 0,0,0 μεσα στην else. Εδω παρατηρουσαμε καποια προβληματα αναλογα με την θεση απο την οποια ξεκινουσε ο μετεωριτης (οπως πχ αρνητικο x και αρνητικο y και αρνητικο z) και ετσι πηραμε μια ειδικη περιπτωση για καθε ενα απο αυτα τα προβληματα (4) . Μετα αλλαζουμε με τις ιδιες τιμες τις μεταβλητες Meteor_x, Meteor_y, Meteor_z γιατι τις χρειαζομαστε για να υπολογισουμε μετα την συγκρουση(η translate κανει την μετατοπιση αλλα δεν αλλαζει τις Meteor_x, Meteor_y, Meteor_z που ορισαμε εμεις προφανως) . Τελος ελεγχουμε αν οι συντεταγμενες x,y,z του Meteor ειναι αναμεσα στις τιμες [-1,1] οπου πρακτικα σημαινει οτι εφτασε στο κεντρο του ηλιου αρα και την μετατοπιζουμε στο 0,0,0 ακριβως ωστε να ειναι σωστη η επομενη μετατοπιση που θα γινει στην θεση της καμερας, σταματαμε την κινηση (MeteorMovement = 0) και κανουμε το able_to_press_space =1 για να μπορει ο χρηστης να

ξαναπατησει το space.

```
224    if (MeteorMovement == 1 && collision == 0) {
225        checkForCollision(Meteor_x, Meteor_y, Meteor_z);
226        if (Meteor_x < 0 && Meteor_y < 0 && Meteor_z < 0) {
227
228            ModelMatrix3 = glm::translate(ModelMatrix3, glm::vec3(speedOfX, speedOfX * ratioX2Y, speedOfX * ratioX2Z));
229        }
230        else if (Meteor_x < 0 && Meteor_y > 0 && Meteor_z < 0) {
231            ModelMatrix3 = glm::translate(ModelMatrix3, glm::vec3(speedOfX, speedOfX * ratioX2Y, speedOfX * ratioX2Z));
232        }
233        else if (Meteor_x < 0 && Meteor_y < 0 && Meteor_z > 0) {
234            ModelMatrix3 = glm::translate(ModelMatrix3, glm::vec3(speedOfX, speedOfX * ratioX2Y, speedOfX * ratioX2Z));
235        }
236        else if (Meteor_x < 0 && Meteor_y > 0 && Meteor_z > 0) {
237            ModelMatrix3 = glm::translate(ModelMatrix3, glm::vec3(speedOfX, speedOfX * ratioX2Y, speedOfX * ratioX2Z));
238        }
239        else {
240            ModelMatrix3 = glm::translate(ModelMatrix3, glm::vec3(-speedOfX, -speedOfX * ratioX2Y, -speedOfX * ratioX2Z));
241        }
242
243        if (Meteor_x > 0) {
244            Meteor_x -= speedOfX;}
245        else {
246            Meteor_x += speedOfX;}
247
248        if (Meteor_x > 0) {
249            Meteor_y -= speedOfX * ratioX2Y;}
250        else {
251            Meteor_y += speedOfX * ratioX2Y;}
252
253        if (Meteor_x > 0) {
254            Meteor_z -= speedOfX * ratioX2Z;}
255        else {
256            Meteor_z += speedOfX * ratioX2Z;}
257
258        if (Meteor_x >= -1 && Meteor_x <= 1 && Meteor_y >= -1 && Meteor_y <= 1 && Meteor_z >= -1 && Meteor_z <= 1) {
259            ModelMatrix3 = glm::translate(ModelMatrix3, glm::vec3(-Meteor_x,-Meteor_y,-Meteor_z));
260            MeteorMovement = 0;
261            able_to_press_space = 1;
262        }
263
264
265    }
```

Η συναρτηση checkForCollision(M_x,M_y,M_z) δινει στις μεταβλητες Planet_x, Planet_y, Planet_z τις συντεταγμενες του κεντρου του πλανητη (τις οποιες παιρνουμε απο τον ModelMatrix της) . Καναμε print ολες τις τιμες του ModelMatrix2 και πως αυτες μεταβαλλονται σε καθε μετατοπιση και συνειδητοποιησαμε οτι οι μονες που αλλαζουν ειναι οι [0][0],[0][2],[2][0],[2][2] απο τις οποιες κρατησαμε τις cos(x) και sin(x) (τα cos και sin ανηκουν στο [-1,1]). Η x (cos) κυμαινεται στο

25* [-1,1] δηλαδη στο [-25,25] η y ειναι σταθερη στο 0 και η z (sin) κυμαινεται στο 25* [-1,1] δηλαδη στο [-25,25]. Κρατησαμε μεσα στον κωδικα και τις printf για την θεση του πλανητη μετα απο καθε μετατοπιση. Μετα υπολογισαμε την αποσταση του κεντρου του πλαντητη απο τον μετεωριτη με τον τυπο :

$$\text{Distance} = ((x1-x2)^2 + (y1-y2)^2 + (z1-z2)^2)^{0.5}$$

Αν αυτη η αποσταση ειναι μικροτερη ή ιση του αθροισματος των ακτινων τους τοτε κανουμε το collision =1 δηλαδη σταματαει η κανονικη κινηση του πλανητη που ειδαμε παραπανω , σταματαει η κανονικη κινηση του μετεωριτη γιατι παρολο που το MeteorMovement ειναι ακομα 1 το collision πλεον ειναι 1 αρα δεν μπαινει μεσα στην if. Πλεον ομως μπαινουμε στην if(collision ==1) οπου αλλαζει την κινηση των 2 αντικειμενων και τους στελνει προς 2 διαφορετικες κατευθυνσεις.

```
70  bool checkForCollision(float M_x,float M_y, float M_z) {
71
72      Planet_x = 25*ModelMatrix2[0][0];
73      Planet_y = 0.0;
74      Planet_z = -25*ModelMatrix2[2][0];
75
76      //printf("Planet x = %f ", Planet_x);
77      //printf("Planet y = %f ", Planet_y);
78      //printf("Planet z = %f ", Planet_z);
79      //printf("\n");
80      distance_of_planets = pow((pow((Planet_x - M_x),2) + (pow((Planet_y - M_y), 2)) + pow((Planet_z - M_z), 2)),0.5);
81      if (distance_of_planets <= 7){
82          collision = 1;
83
84      }
85
86      return true;
87
88  }
89
```

```
202     if (collision == 1) {
203         ModelMatrix2 = glm::translate(ModelMatrix2, glm::vec3(0.1, 0.1, 0));
204         ModelMatrix3 = glm::translate(ModelMatrix3, glm::vec3(-0.1,0,-0.1));
205
206     }
```