ΑΝΑΦΟΡΑ

Γραφικά Υπολογιστών και Συστήματα Αλληλεπίδρασης

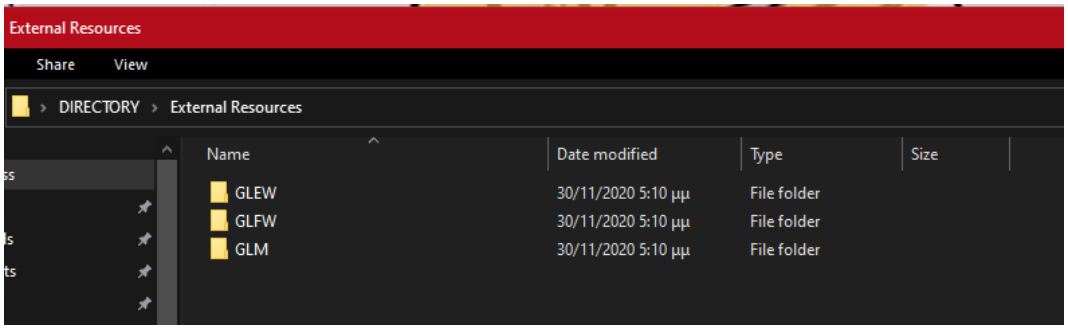
1η Άσκηση

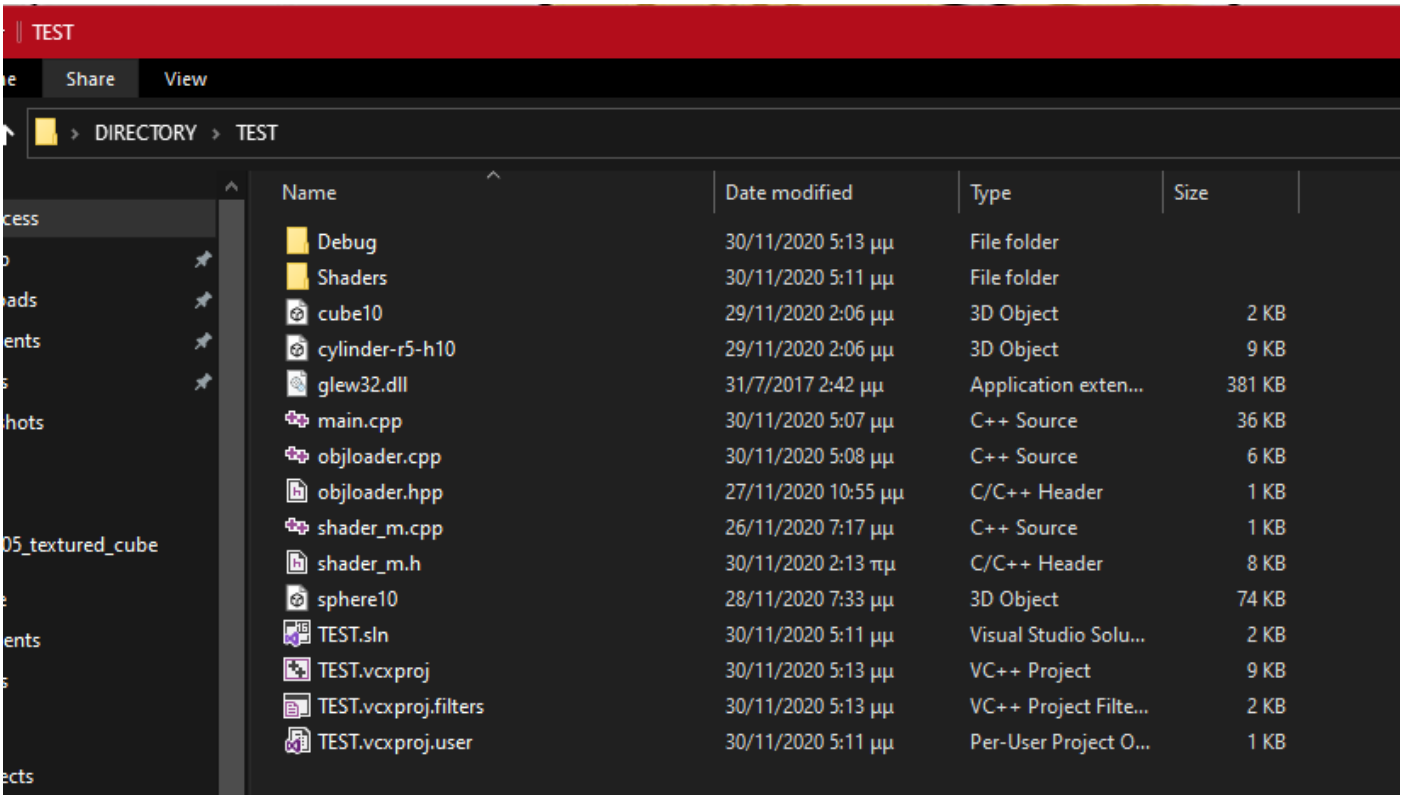
Ονοματεπώνυμο : Σκαρλάτου Δανάη

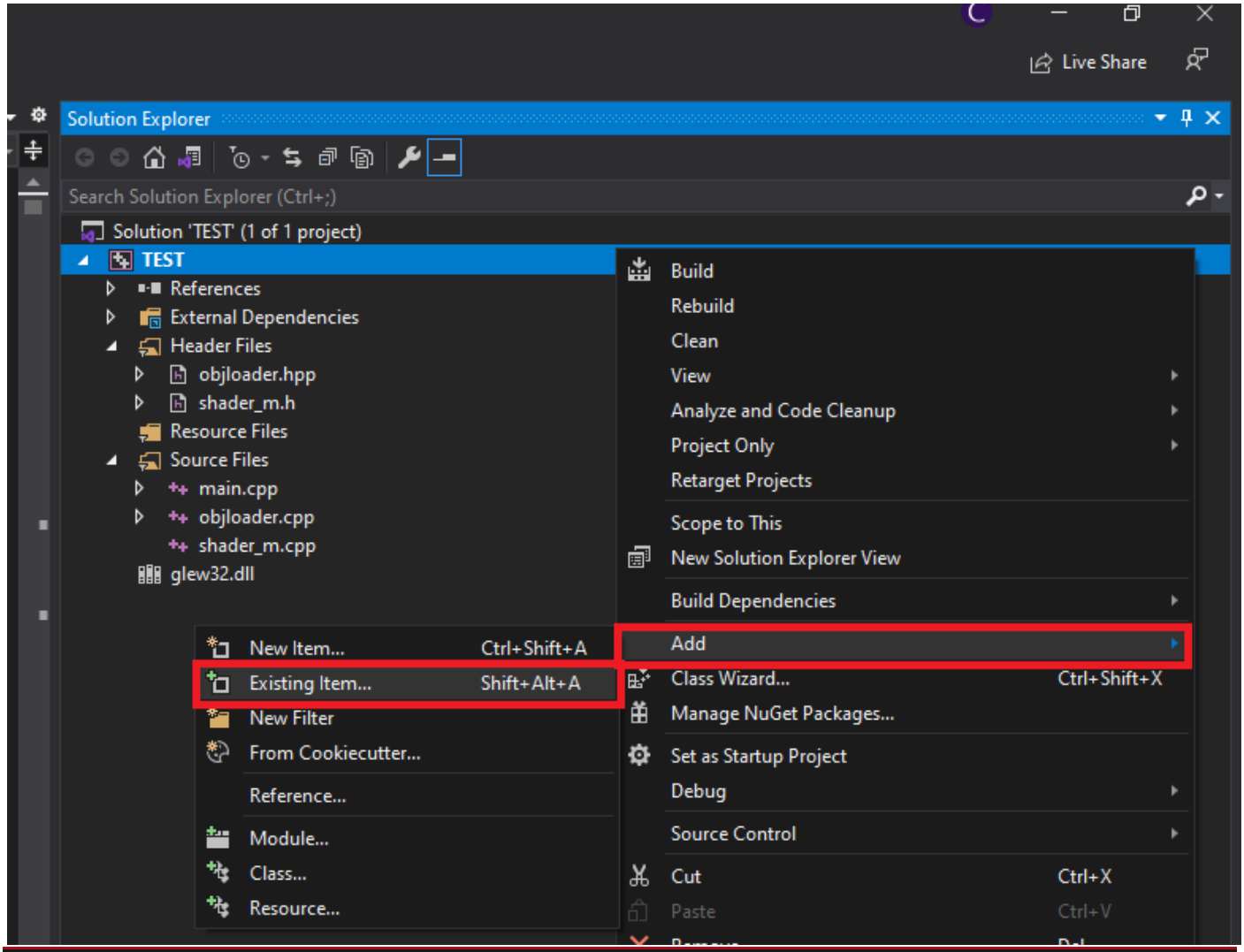
A.M. : 2908

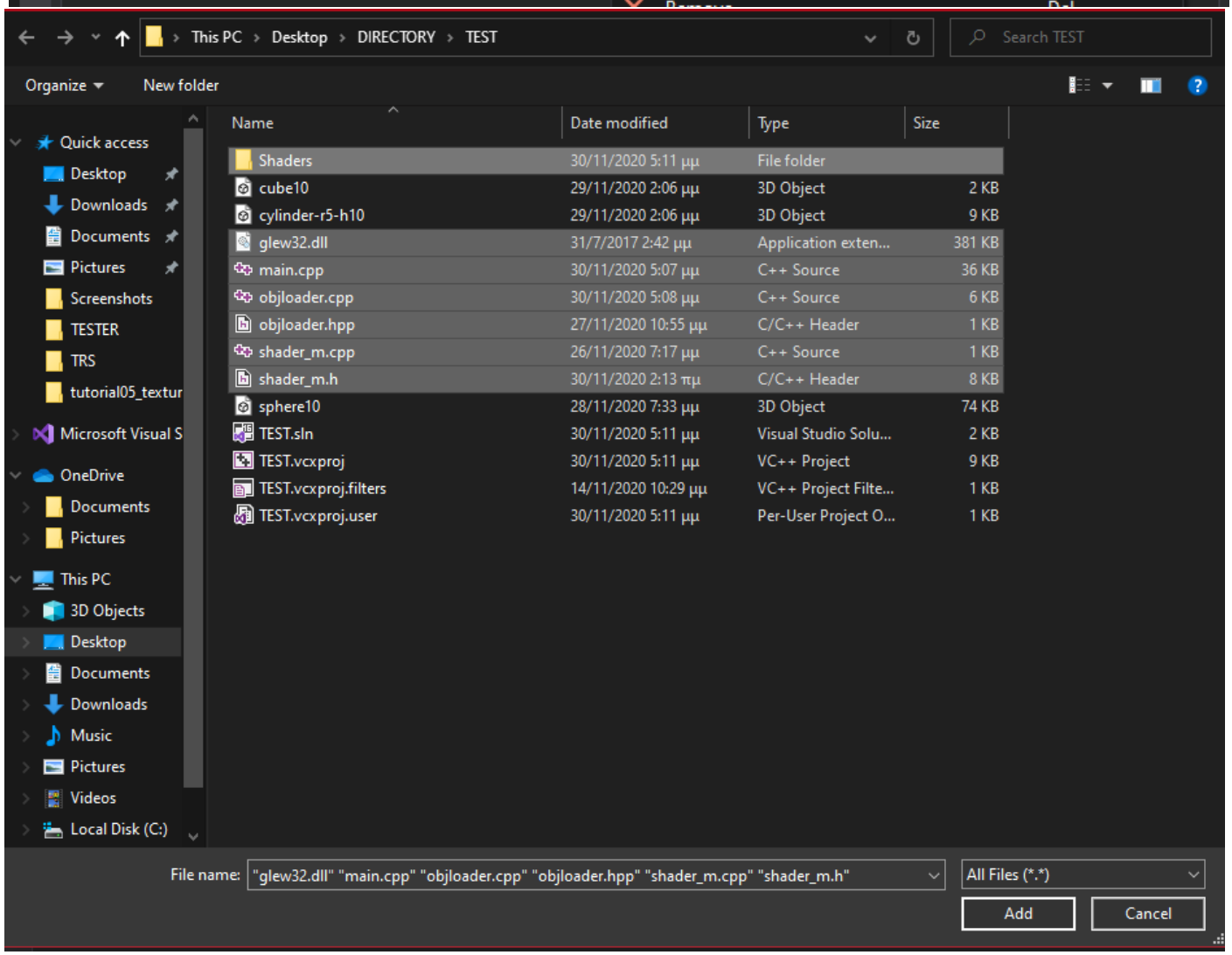
Για να τρέξει η εργασία στον υπολογιστή σας

1. Δημιουργία φακέλου (έστω DIRECTORY)
2. Μέσα στο νέο φάκελο, δημιουργία νέου φακέλου, πρέπει να έχει το όνομα External Resources
3. Ο External Resources έχει τις βιβλιοθήκες GLFW, GLEW, GLM
4. Δημιουργία νέου πρότζεκτ στο Visual Studio 2019, GLFW-GLEW-GLM template, path /your_path/ DIRECTORY. Έστω ότι το ονομάζουμε TEST
5. Extract το αρχείο που στάλθηκε στο /your_path/ DIRECTORY / TEST
6. Στο Visual Studio 2019, διαγραφή οποιουδήποτε αρχείου .cpp υπάρχει μέσα (δημιουργείται από το template)
7. Στο Visual Studio 2019, δεξί κλικ στο solution TEST, add existing item, επιλογή όλων των αρχείων εκτός από τα .obj.
8. Build
9. Run






Live Share
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'TEST' (1 of 1 project)
TEST
References
External Dependencies
Header Files
objloader.hpp
shader_m.h
Resource Files
Source Files
main.cpp
objloader.cpp
shader_m.cpp
glew32.dll
Build
Rebuild
Clean
View
Analyze and Code Cleanup
Project Only
Retarget Projects
Scope to This
New Solution Explorer View
Build Dependencies
Add
Class Wizard... Ctrl+Shift+X
Manage NuGet Packages...
Set as Startup Project
Debug
Source Control
Cut Ctrl+X
Paste Ctrl+V
New Item... Ctrl+Shift+A
Existing Item... Shift+Alt+A
New Filter
From Cookiecutter...
Reference...
Module...
Class...
Resource...


This PC › Desktop › DIRECTORY › TEST
Search TEST
Organize
New folder
Quick access
Desktop
Downloads
Documents
Pictures
Screenshots
TESTER
TRS
tutorial05_textur
Microsoft Visual S
OneDrive
Documents
Pictures
This PC
3D Objects
Desktop
Documents
Downloads
Music
Pictures
Videos
Local Disk (C:)
Name
Date modified
Type
Size
Shaders 30/11/2020 5:11 μμ File folder
cube10 29/11/2020 2:06 μμ 3D Object 2 KB
cylinder-r5-h10 29/11/2020 2:06 μμ 3D Object 9 KB
glew32.dll 31/7/2017 2:42 μμ Application exten... 381 KB
main.cpp 30/11/2020 5:07 μμ C++ Source 36 KB
objloader.cpp 30/11/2020 5:08 μμ C++ Source 6 KB
objloader.hpp 27/11/2020 10:55 μμ C/C++ Header 1 KB
shader_m.cpp 26/11/2020 7:17 μμ C++ Source 1 KB
shader_m.h 30/11/2020 2:13 πμ C/C++ Header 8 KB
sphere10 28/11/2020 7:33 μμ 3D Object 74 KB
TEST.sln 30/11/2020 5:11 μμ Visual Studio Solu... 2 KB
TEST.vcxproj 30/11/2020 5:11 μμ VC++ Project 9 KB
TEST.vcxproj.filters 14/11/2020 10:29 μμ VC++ Project Filte... 1 KB
TEST.vcxproj.user 30/11/2020 5:11 μμ Per-User Project O... 1 KB
File name: "glew32.dll" "main.cpp" "objloader.cpp" "objloader.hpp" "shader_m.cpp" "shader_m.h"
All Files (*.*)
Add
Cancel

Υλοποίηση Εργασίας

(i) Για τη δημιουργία του παραθύρου χρησιμοποίησα τις διαφάνειες του εργαστηρίου. Για να ορίσω το αρχικό μέγεθος του παραθύρου χρησιμοποιώ δύο μεταβλητές(SCR_WIDTH, SCR_HEIGHT), τις οποίες όρισα global για ευκολότερη αλλαγή μεγέθους στο μέλλον. Πρόσθεσα την επιλογή για resize window. Αν γίνει resize, αλλάζει το scale των αντικειμένων μέσα στο παράθυρο. Αυτό σημαίνει ότι αν κάνω το παράθυρο ορθογώνιο, τότε και ο κύβος μέσα θα είναι ορθογώνιος.

Η τιμή ALPHA του κύβου είναι επίσης global και ίση με 0.4.

Η δημιουργία του κύβου έχει γίνει χρησιμοποιώντας vertices. Ο κύβος εκτείνεται από το (0,0,0) μέχρι το (100,100,100).

Για να φανεί ότι ο κύβος είναι κύβος κι όχι εξάγωνο απαιτείται φωτισμός. Έτσι, ο κύβος χρησιμοποιεί shader με φωτισμό.

Η επιλογή του χρώματος του κύβου γίνεται τυχαία. Επιλέγονται 3 τυχαίοι αριθμοί στο διάστημα [0.0, 1.0] και αυτοί ορίζουν τις rgb τιμές του κύβου.

Tutorial και links για αυτό το κομμάτι

https://www.youtube.com/watch?v=uTJRjIjUnoM&t=1519s&ab_channel=LewisMarlow (transparency)

https://www.glfw.org/docs/3.3.2/window_guide.html (window title)

https://learnopengl.com/Lighting/Basic-Lighting (lighting cube)

https://www.glfw.org/docs/3.3/input_guide.html (Keyboard input)

(ii) Για αυτό το ερώτημα χρησιμοποιήθηκαν μοντέλα από το blender με d = 10m.
Πατώντας <SPACE> δημιουργείται ένα νέο τυχαίο αντικείμενο με τυχαίο χρώμα, κατεύθυνση και μέγεθος. Εάν ο χρήστης πατήσει <SPACE> παρατεταμένα θα δημιουργηθεί μόνο ένα νέο σχήμα. Για δημιουργία π.χ. 3 σχημάτων πρέπει να πατήσουμε 3 φορές το <SPACE>.
Στο ερώτημα αυτό μας ζητείται το νέο αντικείμενο να κάνει spawn στο σημείο (0,0,0). Αυτό όμως, π.χ. για ένα κύβο με πλευρά 10μ, σημαίνει ότι το κέντρο του κύβου θα είναι στο (0,0,0) και κάποιο μέρος του θα είναι εκτός του κύβου της σκηνής. Για αυτό το λόγο επέλεξα το νέο σχήμα να μην ξεκινάει από το (0,0,0) αλλά να ξεκινάει από το σημείο (d/2, d/2, d/2). Έτσι δημιουργείται η ψευδαίσθηση ότι το νέο σχήμα ξεκινάει από την αρχή των αξόνων (0,0,0). Το ίδιο ισχύει και για τα άλλα δύο σχήματα.
Για την σύγκρουση με τα τοιχώματα του κύβου της σκηνής, δημιούργησα συνάρτηση η οποία ελέγχει αν κάποιο αντικείμενο έχει ακουμπήσει το 0 ή το 100 σε κάθε ένα από τους άξονες ξεχωριστά. Εάν η συντεταγμένη του σχήματος ακουμπήσει στο 0 ή στο 100 τον άξονα X, τότε αντιστρέφω το vx (vx = -vx). Αντίστοιχα και για τους άλλους άξονες. Λαμβάνοντας όμως υπόψη ότι το κέντρο του κύβου είναι αυτό που ακουμπάει στην ουσία τον άξονα, πρέπει να λάβω υπόψη το μέγεθος του σχήματος για να φαίνεται πιο ρεαλιστικό και να μην βγαίνει κανένα μέρος του σχήματος εκτός του κύβου της σκηνής. Έτσι, αντί να ελέγξω ότι ακουμπάει το 0, ελέγχω ότι ακουμπάει το 0 + d/2, αντίστοιχα για το 100 ελέγχω το 100 - d/2.
Τα σχήματα που δημιουργούνται τα αποθηκεύω σε 3 δισδιάστατους global ‘πίνακες’, ένας πίνακας για κυλίνδρους, ένας για κύβους και ένας για σφαίρες. Πρέπει να κρατάω τη θέση, μέγεθος, κατεύθυνση και χρώμα του κάθε αντικειμένου. Οι πίνακες μου έχουν την εξής μορφή.

	RGB	POSITION	SCALE	VECTOR
Cylinder 0				
Cylinder 1				

Tutorial και links για αυτό το κομμάτι

https://exploratoria.github.io/exhibits/mechanics/elastic-collisions-in-3d/#:~:text=A bunch of bouncing balls conserved and does not change. (elastic collision visualization, math and code)

(iii) Για δημιουργία σφαίρας ακτίνας 15μ, χρησιμοποιώ το ίδιο μοντέλο σφαίρα από το blender, το οποίο έχει d = 10 μ, r = 5 μ. Για να γίνει ακτίνα 15μ, το κάνω scale(3.0), 5 x 3 = 15m. Η σφαίρα ξεκινάει από το (50,50,50) και έχει χρώμα (1,0,0).
Η κίνηση της σφαίρας γίνεται με τα πλήκτρα <UP><DOWN><LEFT><RIGHT><ADD><SUBTRACT>. Απαιτείται να υπάρχουν μεμονωμένα αυτά τα πλήκτρα στο πληκτρολόγιο, τα μικρά πληκτρολόγια που δεν έχουν μεμονωμένα τα πλήκτρα +,- δεν θα μπορούν να μετακινήσουν την σφαίρα στον Z άξονα.
Η κίνηση γίνεται λαμβάνοντας υπόψη την ακτίνα της, έτσι ώστε η σφαίρα να μένει πάντα μέσα στο κύβο της σκηνής.
Η σύγκρουση με τα μικρά στοιχειώδη αντικείμενα έγινε όπως στο παραπάνω ερώτημα. Δηλαδή λαμβάνοντας υπόψη τις θέσεις των αντικειμένων και τα μεγέθη τους, και σε περίπτωση σύγκρουσης αλλάζει η κατεύθυνση του μικρού αντικειμένου, ενώ η σφαίρα δεν επηρεάζεται.
Μερικά προβλήματα που παρατηρήθηκαν είναι το εξής: Εάν σπρώξουμε το μικρό αντικείμενο με τη σφαίρα στη γωνία του κύβου, το μικρό αντικείμενο 'τρεμοπαίζει' προσπαθώντας να ξεφύγει. Ακόμα, εάν η σφαίρα είναι πιο γρήγορη από το αντικείμενο, τότε μπορούμε να παγιδεύσουμε το μικρό αντικείμενο εντός της σφαίρας. Το μικρό αντικείμενο θα κολλήσει μέχρι να πάρουμε τη σφαίρα από πάνω του.

Tutorial και links για αυτό το κομμάτι

https://www.glfw.org/docs/3.3/input_guide.html
https://www.glfw.org/docs/3.3/group__keys.html

(iv) Η κίνηση της κάμερας γίνεται με τα πλήκτρα <W><S><A><D><E><X> . Η κάμερα κοιτάει συνεχώς στο κέντρο του κύβου (50,50,50). Αρχικά τοποθετείται στο σημείο (180,150,-130), η επιλογή αυτού του σημείου έγινε για να φαίνεται ο κύβος καλά και τα αντικείμενα τα οποία περιέχει. Κινώντας την κάμερα μπορούμε να δούμε τον κύβο από διάφορες γωνίες.

Tutorial και links για αυτό το κομμάτι

https://learnopengl.com/Getting-started/Camera

Bonus

(i) Σε κάθε σύγκρουση τα μικρά στοιχειώδη αντικείμενα αλλάζουν χρώμα. Η επιλογή του νέου χρώματος γίνεται τυχαία.

(ii) Με τα πλήκτρα '<', '>' αλλάζει η ταχύτητα όλων των μικρών αντικειμένων μέσα στο κύβο. Δεν υπάρχει αρνητική ταχύτητα, έτσι, εάν πατήσουμε παρατεταμένα το '<' τα αντικείμενα θα σταματήσουν να κινούνται καθώς η ταχύτητα τους έχει μηδενιστεί. Εάν όμως μηδενιστεί η ταχύτητα και το μικρό αντικείμενο ακουμπάει στο τοίχωμα του κύβου ή τη μεγάλη σφαίρα, τότε θα αλλάζει συνεχώς χρώμα.

(iii) Προσθήκη σύγκρουσης αντικειμένων μεταξύ τους. Σε αυτό το κομμάτι, εάν κάνουμε spawn 2 αντικείμενα πολύ γρήγορα(δηλαδή, δεν δώσουμε χρόνο να φύγει το 1ο) τότε τα δύο αυτά αντικείμενα συγκρούονται από την αρχή μεταξύ τους και φαίνεται σαν να έχει κολλήσει το ένα μέσα στο άλλο. Σε αυτό το μέρος, επέλεξα να μην τους αλλάζω χρώματα όταν χτυπάνε μεταξύ τους, καθώς πιστεύω θα ήταν αρκετά άσχημο στο μάτι.

Μερικές πληροφορίες επάνω στη λειτουργία του κώδικα

```
void KeysCheck();
```

Ελέγχει εάν πατήθηκε κάποιο πλήκτρο και ανανεώνει το ανάλογο πεδίο. Ανανεώνει τη θέση της κάμερας, τη θέση της σφαίρας και την ταχύτητα των μικρών αντικειμένων

```
void spawnShape();
```

Όταν καλείται, δημιουργεί ένα νέο τυχαίο σχήμα με τυχαία κατεύθυνση, χρώμα και μέγεθος. Ανάλογα τι σχήμα δημιούργησε το προσθέτει στο κατάλληλο πίνακα μαζί με τις τιμές του. Επίσης δίνει και την αρχική θέση του μικρού αντικειμένου.

```
void key_callback(GLFWwindow* window, int key, int scancode, int action, int mods);
```

Είναι για να χειρίζεται μόνο ένα πλήκτρο, το SPACE. Έτσι ώστε να μην μπορούμε να δημιουργούμε πολλαπλά αντικείμενα όταν πατάμε παρατεταμένα το πλήκτρο

```
void framebuffer_size_callback(GLFWwindow* window, int width, int height);
```

Υπεύθυνη για αλλαγή μεγέθους του αρχικού παραθύρου και όλα όσα περιέχει

```
void checkOnAxis(int shape, double scale, int index);
```

Σε περίπτωση που ένα αντικείμενο βγει εκτός του κύβου, τότε θα τοποθετηθεί πίσω στο κύβο.

Παράμετροι: σχήμα (1 για κύβο, 2 για σφαίρα, 3 για κύλινδρο), μέγεθος, θέση στον αντίστοιχο πίνακα

```
void makeCube();      void makeSphere();    void makeCylinder();
```

Δημιουργούν το αντίστοιχο σχήμα όταν καλούνται.

```
float getRandomScale();
```

Επιστρέφει έναν αριθμό από τον πίνακα [0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9]. Για να γίνει σωστά το scale και έτσι η διάμετρος, ύψος, πλευρά των αντικειμένων να έχουν ακέραια τιμή.

```
glm::vec3  movement(float scale, int sh, glm::vec3 p, glm::vec3 v, glm::vec3 c, int i);
```

Είναι υπεύθυνη για την κίνηση των μικρών αντικειμένων και την σύγκρουση τους με τα τοιχώματα του κύβου, την μεγάλη σφαίρα και την μεταξύ τους σύγκρουση. Παράμετροι: το μέγεθος του σχήματος, το σχήμα (1 για κύβο, 2 για σφαίρα, 3 για κύλινδρο), θέση, κατεύθυνση, χρώμα, θέση στον αντίστοιχο πίνακα.

```
int main();
```

Πριν το render loop ρυθμίζουμε τις βιβλιοθήκες και το παράθυρο. Επίσης γίνεται αρχικοποίηση των Shaders, Buffers και το τυχαίο χρώμα του κύβου.

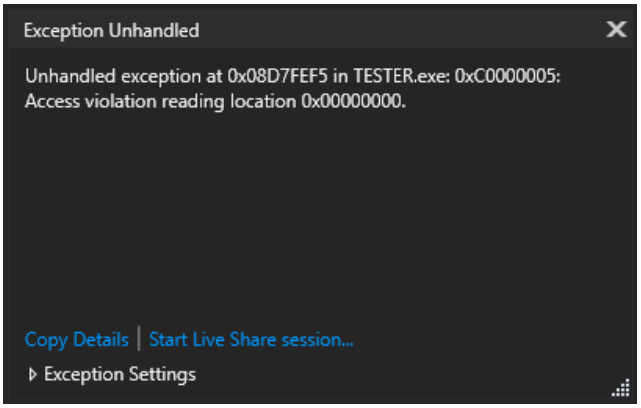
Στο render loop κάνουμε render την μεγάλη σφαίρα και τον κύβο της σκηνής. Για τα μικρά αντικείμενα υπάρχει ένα ενφωλευμένο loop το οποίο είναι υπεύθυνο για το render όλων των μικρών αντικειμένων. Οι θέσεις, χρώματα, κατευθύνσεις τους αλλάζουν συνεχώς και για αυτό πρέπει να αποθηκεύονται συνεχώς στους αντίστοιχους global πίνακες, όπως και να ανανεώνονται οι αντίστοιχες τιμές των shaders.

(extra)

Το Debug Console εμφανίζει πληροφορίες για τα αντικείμενα που δημιουργούνται

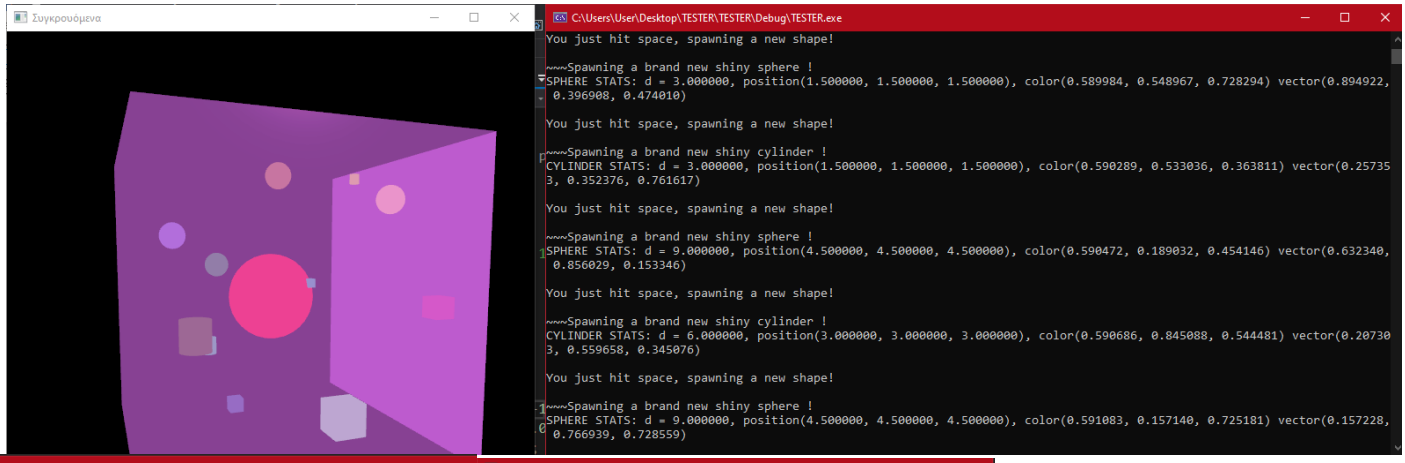
Προβλήματα που εμφανίστηκαν και παρατηρήσεις

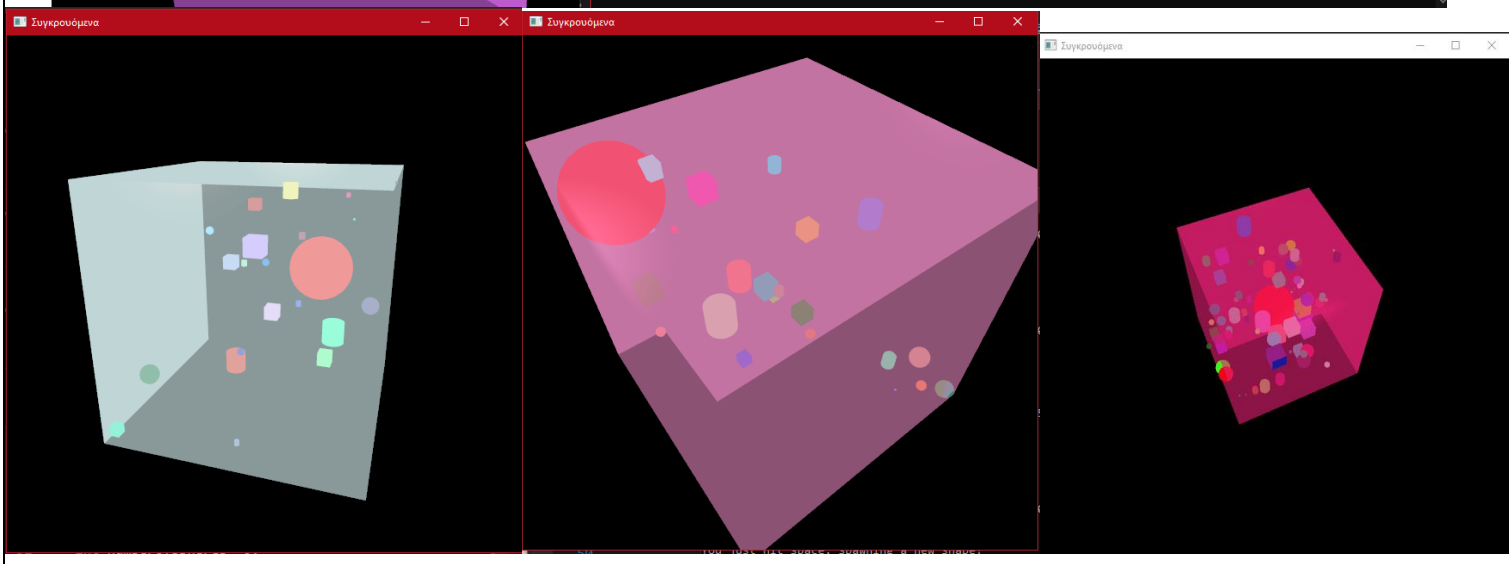
1. Εάν αφήσουμε το παράθυρο πολύ ώρα ανοιχτό ή το υπερφορτώσουμε με αντικείμενα βγαίνει error



2. Μερικές φορές κατά τη σύγκρουση τα αντικείμενα μπορεί να κολλήσουν το ένα μέσα στο άλλο και να τρεμοπαίζουν. Εάν κολλήσουν μέσα στη μπάλα ή στο τοίχωμα του κύβου αλλάζουν χρώμα συνεχώς.
3. Καθώς δεν έχω βάλει πάνω όριο στη ταχύτητα, εάν η ταχύτητα αναπτυχθεί πολύ τότε μπορεί τα αντικείμενα να βγουν εκτός του κύβου. Αυτό διορθώνεται χαμηλώνοντας την ταχύτητα.

Μερικά στιγμιότυπα





Disclaimer

Τα αρχεια shader_m.hpp, shader_m.cpp είναι από αυτή την ιστοσελίδα, με μία πολύ μικρή αλλαγή (κενό constructor)

https://learnopengl.com/code_viewer_gh.php?code=includes/learnopengl/shader.h

Τα αρχεια objloader.hpp, objloader.cpp είναι από αυτή την ιστοσελίδα, με μία πολύ μικρή αλλαγή. (#pragma warning(disable : 4996))
http://www.opengl-tutorial.org/beginners-tutorials/tutorial-7-model-loading/