Γραφικά Υπολογιστών και Συστήματα Αλληλεπίδρασης

Γ Μέρος -Σύνολο Προγραμματιστικών Ασκήσεων

***Μέλη ομάδας***  :

*Μανίκας Ελευθέριος Μάριος ,4723*

*Φωτόπουλος Στέφανος ,4829*

Ιωάννινα,03/12/2022

Contents

[Γ Μέρος -Σύνολο Προγραμματιστικών Ασκήσεων 1](#_Toc120981436)

[1.Περιγραφή της εργασίας 3](#_Toc120981437)

[Ερώτημα 1 3](#_Toc120981438)

[Ερώτημα 2 4](#_Toc120981439)

[Ερώτημα 3 6](#_Toc120981442)

[Ερώτημα 4 7](#_Toc120981445)

[2.Πληροφορίες σχετικά με την υλοποίηση 12](#_Toc120981446)

[3.Σύντομη αξιολόγηση λειτουργίας ομάδας 12](#_Toc120981447)

[4.Αναφορές-Πηγές 12](#_Toc120981448)

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).



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.



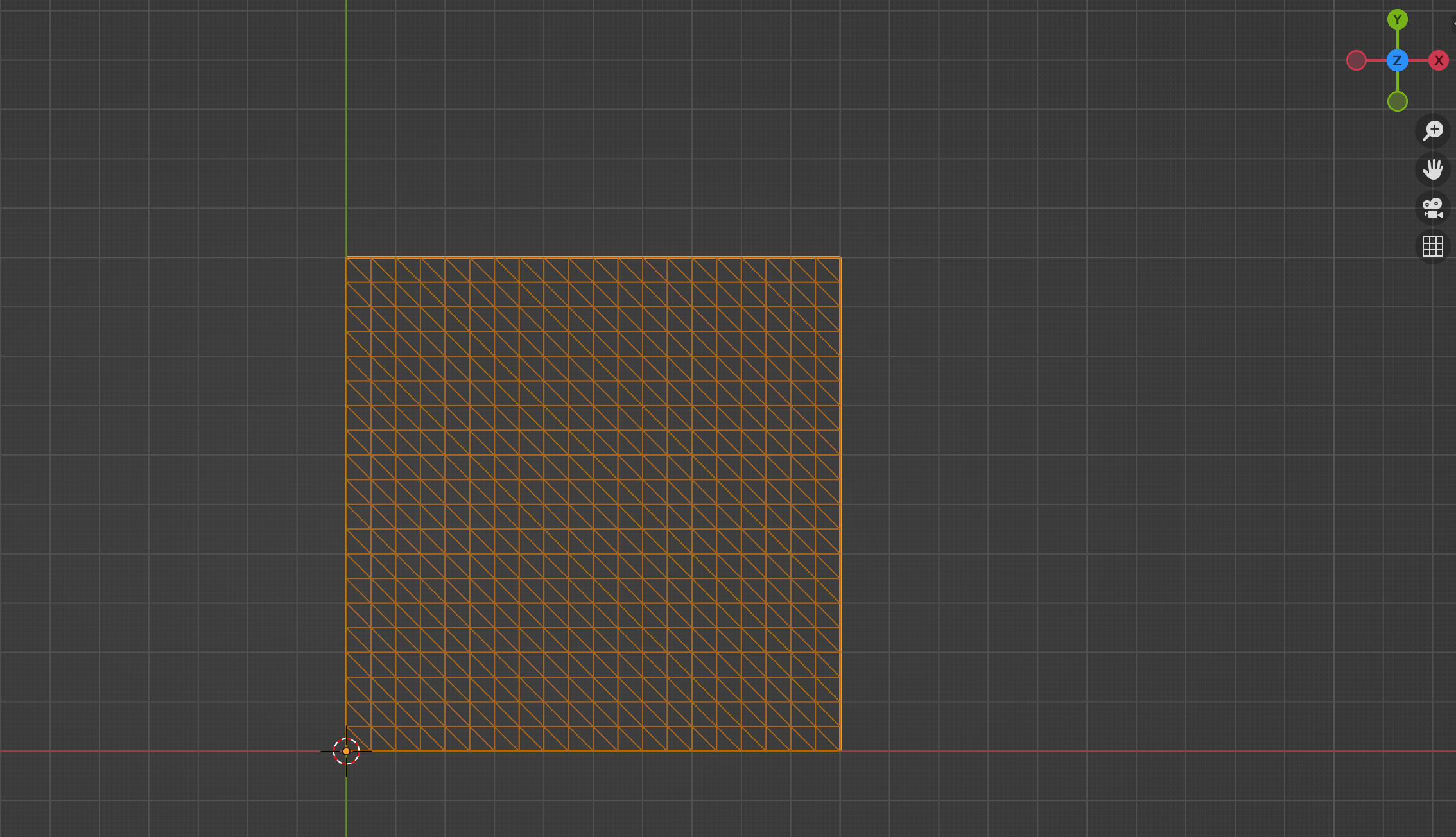
Εικόνα 2

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 Χ.



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.

Line chart

Description automatically generated

Εικόνα

Text

Description automatically generated

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.

Text

Description automatically generated

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.

Text

Description automatically generated

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.

Text

Description automatically generated

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.

Text

Description automatically generated

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.

A screenshot of a computer

Description automatically generated with medium confidence

Εικόνα 11

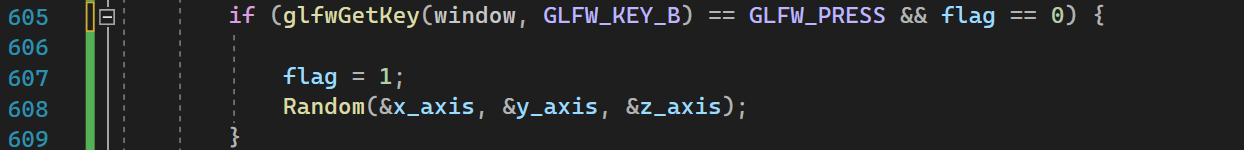
Στην εικόνα 12 κάνουμε Bind με τα Buffers που έχουμε προαναφέρει για το grid και το ζωγραφίζουμε .

Text

Description automatically generated

Εικόνα 12

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



Εικόνα 13

Όταν η τιμή του flag μας είναι 1(δηλαδή πατηθεί το Β) 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.

Text

Description automatically generated

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 Β.

Text

Description automatically generated

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.

Text

Description automatically generated

Εικόνα 16

Text

Description automatically generated

Εικόνα

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

Text

Description automatically generated

Εικόνα

## 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->Ιnput->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.