ΑΝΑΦΟΡΑ ΠΡΟΓΡΑΜΜΑΤΙΣΤΙΚΗΣ ΑΣΚΗΣΗΣ Ι

ΟΝΟΜΑΤΕΠΩΝΥΜΑ:

Γεωργούλας Δημοσθένης AM: 4039

Στεργίου Βασίλειος AM: 4300

Το παραδωταίο πρόγραμμα λειτουργεί με βάση όσα απαιτούνται σε κάθε ερώτημα της προγραμματιστικής άσκησης. Πιο συγκεκριμένα, αρχικά ανοίγει ένα παράθυρο διαστάσεων 600 x 600, μαύρου χρώματος με τίτλο “ Συγκρουόμενα”. Στο κέντρο της σκηνής μας εμφανίζεται ο διαφανής κύβος που εκτείνεται από το σημείο (0,0,0) έως το σημείο (100,100,100) στο παγκόσμιο σύστημα συντεταγμένων, καθώς και η κόκκινη αδιαφανής σφαίρα που βρίσκεται στο κέντρο του κύβου. Με το πάτημα του πλήκτρου escape η εφαρμογή τερματίζει.

Σχετικά με την χρήση της κάμερας, η κάμερα κοιτάει πάντα στο κέντρο του κύβου και για την κίνησή της χρησιμοποιούνται τα πλήκτρα D και A για περιμετρική κίνηση δεξιά και αριστερά του κύβου αντίστοιχα, τα πλήκτρα W και S για την κατακόρυφη κίνηση της κάμερας προς τα πάνω και προς τα κάτω αντίστοιχα και τα πλήκτρα E και X για να πλησιάζουμε και να απομακρύνουμε την κάμερα.

Μετά την κατασκευή του κύβου, δημιουργείται η κεντρική σφαίρα και παράλληλα με το πάτημα του πλήκτρου spacebar, δημιουργείται ένα τυχαίο αντικείμενο τυχαίων διαστάσεων και τυχαίου χρώματος, καθώς και ένα τυχαίο διάνυσμα κίνησης του αντικειμένου. Το κάθε αντικείμενο συγκρούεται με τον κύβο και την κεντρική σφαίρα και αναπηδά με γωνία ανάκλασης ίση με την γωνία πρόσκρουσης.

Σχετικά με την κίνηση της κεντρικής σφαίρας, χρησιμοποιούνται το πάνω και κάτω βελάκι για κίνηση στον Y άξονα, το δεξί και το αριστερό βελάκι για κίνηση στον X άξονα και τα <+>, < - > για κίνηση στον Z άξονα.

Επιπρόσθετα, έχει υλοποιηθεί η προσθήκη και αφαίρεση texture με το πάτημα του πλήκτρου T, με την χρήση του οποίου ενεργοποιείται και απενεργοποιείται το texture της σφαίρας.

Από τα Bonus ερωτήματα, έχει υλοποιηθεί η αύξηση και η μείωση της ταχύτητας που κινούνται τα αντικείμενα με τη χρήση των πλήκτρων “ >” και “ < ”.

Καθώς προσθέτουμε αντικείμενα με το πάτημα του spacebar, υπάρχει καθυστέρηση στην κίνηση των αντικειμένων και συνεπώς δεν είναι εμφανές ότι τα αντικείμενα κινούνται με σταθερή ταχύτητα. Στον κώδικα που παραδίδουμε δεν υλοποιήσαμε την σταθερή ταχύτητα καθώς αυξάνεται ο αριθμός των αντικειμένων (κυρίως για τις σφαίρες).

Επεξήγηση του κώδικα

Στην αρχή του κώδικα βάζουμε τις απαραίτητες βιβλιοθήκες της OpenGL καθώς και τις βασικές βιβλιοθήκες για εισαγωγή των shaders (`"common/shader.hpp"`), της χρήσης της κάμερας (`"common/controls.hpp"`) και για την εισαγωγή της εικόνας - texture (`"common/stb_image.h"`).

Ορίζουμε τα stacks με define σε έναν σταθερό αριθμό και τα χρησιμοποιούμε για την κατασκευή της σφαίρας και των κυλίνδρων. Καθώς αυξάνεται ο αριθμός των stacks και των sectors, αυξάνεται και η καθυστέρηση στην κίνηση των αντικειμένων.

Για κάθε είδος αντικειμένου, δημιουργούμε και ένα struct, το οποίο κρατά ένα id που καθορίζει το είδος του αντικειμένου που έχουμε (1 για κύβο, 2 για σφαίρα, 3 για κύλινδρο), το vertexBufferID και το colorBufferID με τα οποία κάνουμε initialize τους vertex buffers και τους color buffers του κάθε αντικειμένου και τους πίνακες vertices και colors που κρατούν τις κορυφές και τα χρώματα των αντικειμένων. Οι πίνακες colors κρατούν τις τιμές RGBA κάθε κορυφής. Επίσης, το struct κρατά και ένα διάνυσμα direction που έχει πληροφορία για την κατεύθυνση του αντικειμένου στον χώρο.

Για κάθε είδος αντικειμένου, δημιουργούμε το αντίστοιχο διάνυσμα που κρατά το αντικείμενο. Το διάνυσμα cubeBuffers κρατά όλα τα structs τύπου cube, το διάνυσμα sphereBuffers κρατά όλα τα structs τύπου sphere και διάνυσμα cylinderBuffers κρατά όλα τα structs τύπου cylinder.

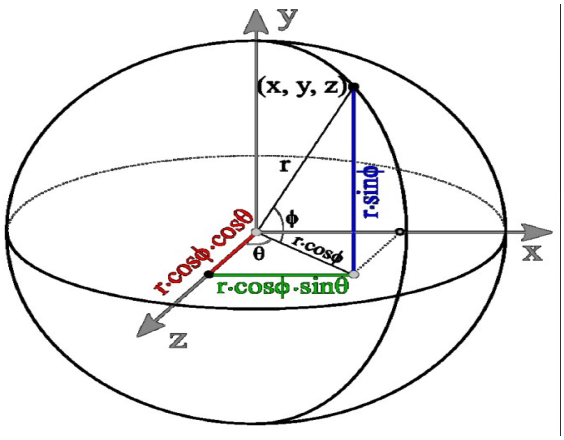
Η συνάρτηση makeCube δέχεται ως όρισμα έναν πίνακα και αντιγράφει σε αυτόν τις κορυφές του κύβου.

Η συνάρτηση makeSphere παίρνει ως όρισμα ένα πίνακα και δημιουργεί τα σημεία της σφαίρας. Αυτό επιτυγχάνεται με τον χωρισμό της σφαίρας σε τετράπλευρα και ύστερα σε 2 τρίγωνα. Χωρίζουμε το τόξο που δημιουργεί ο κατακόρυφος άξονας σε ίσα μέρη, όσα και τα stacks και χωρίζουμε την περίμετρο της σφαίρας σε ίσα μέρη όσα και τα sectors. Η γωνία φ αφορά τον προσδιορισμό του σημείου κατά μήκος των stacks και η γωνία θ για κίνηση κατά μήκος των sectors και κινούμενοι στην περίμετρο της σφαίρας, υπολογίζουμε τα αντίστοιχα τετράπλευρα.
Οι τύποι για τον προσδιορισμό των σημείων είναι:

$$x = (r * cos\varphi) * sin\theta$$

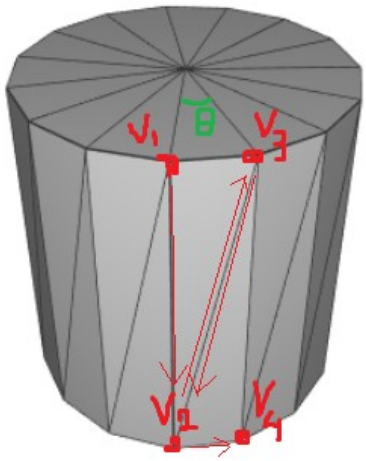
$$y = r\,sin\varphi$$

$$z = (r * cos\varphi) * cos\theta$$



Στην συνέχεια, τις συντεταγμένες των σημείων τις βάζουμε στον πίνακα.

Η συνάρτηση makeCylinder παίρνει ως όρισμα έναν πίνακα και τον γεμίζει με τις συντεταγμένες των σημείων του κυλίνδρου. Πιο συγκεκριμένα, δημιουργούμε στην αρχή τον πάνω κύκλο σε ύψος Η, του οποίου τα σημεία υπολογίζονται παρόμοια με της σφαίρας αλλά στις 2 διαστάσεις (x,y και z σταθερό). Για τον ενδιάμεσο κομμάτι, το χωρίζουμε σε τετράπλευρα και ύστερα σε τρίγωνα και υπολογίζουμε τα σημεία v1,v2,v3,v4.



Τέλος, υπολογίζουμε τον κάτω κύκλο με τον ίδιο τρόπο με τον πάνω σε ύψος Η =0.

Η συνάρτηση makeTexture παίρνει ως όρισμα ένα πίνακα και τον συμπληρώνει με τις u,v που προκύπτουν από προσαρμογή των σημείων της σφαίρας πάνω στο texture. “Ξεδιπλώνουμε” την σφαίρα και αντιστοιχίζουμε τα σημεία της στα αντίστοιχα του texture με αναλογία 1/360 για το u που προκύπτει και 1/180 για το v που προκύπτει.

Η συνάρτηση makeColorsCube παίρνει ως όρισμα έναν πίνακα με σημεία κύβου και ένα επίπεδο transparency, υπολογίζει 3 τυχαίες τιμές RGB και τις αναθέτει σε κάθε κορυφή, μαζί με το transparency.

Η συνάρτηση makeColorsSphere παίρνει ως όρισμα έναν πίνακα με σημεία σφαίρας, υπολογίζει 3 τυχαίες τιμές RGB και τις αναθέτει σε κάθε κορυφή, μαζί με το transparency =1 καθώς όλες οι σφαίρες έχουν solid χρώμα.

Η συνάρτηση makeColorsMainSphere παίρνει ως όρισμα έναν πίνακα με σημεία της κεντρικής σφαίρας, της βάζει κόκκινο χρώμα και τις αναθέτει σε κάθε κορυφή, μαζί με το transparency =1 καθώς όλες οι σφαίρες έχουν solid χρώμα.

Η συνάρτηση makeColorsCylinder παίρνει ως όρισμα έναν πίνακα με σημεία κυλίνδρου, υπολογίζει 3 τυχαίες τιμές RGB και τις αναθέτει σε κάθε κορυφή, μαζί με το transparency =1 καθώς όλοι οι κύλινδροι έχουν solid χρώμα.

Η συνάρτηση scalematrix παίρνει ως όρισμα έναν πίνακα με τις κορυφές κύβου και τις κάνει resize σε ένα μέγεθος στο διάστημα [1,10] στο παγκόσμιο σύστημα συντεταγμένων. Κάθε κύβος που δημιουργείται (εκτός του μεγάλου) ξεκινά από το (randomSize,randomSize,randomSize) για αποφυγή συνεχούς σύγκρουσης με τον μεγάλο κύβο κατά την δημιουργία του.

Η συνάρτηση scalematrixSphere παίρνει ως όρισμα έναν πίνακα με τις κορυφές σφαίρας και τις κάνει
resize σε ένα μέγεθος στο διάστημα [1,10] στο παγκόσμιο σύστημα συντεταγμένων. Κάθε σφαίρα που δημιουργείται (εκτός της κεντρικής) ξεκινά από το (randomSize,randomSize,randomSize) για αποφυγή συνεχούς σύγκρουσης με τον μεγάλο κύβο κατά την δημιουργία της.

Η συνάρτηση scaleCylinder παίρνει ως όρισμα έναν πίνακα με τις κορυφές κυλίνδρου και τις κάνει resize σε ένα μέγεθος στο διάστημα [1,10] στο παγκόσμιο σύστημα συντεταγμένων. Κάθε κύλινδρος που δημιουργείται (εκτός της κεντρικής) ξεκινά από το (randomSize,randomSize,randomSize) για αποφυγή συνεχούς σύγκρουσης με τον μεγάλο κύβο κατά την δημιουργία του.

Η συνάρτηση InitObject παίρνει για ένα αντικείμενο τα id των color και vertex buffers, καθώς και τα στοιχεία των ίδιων των buffers (vertexdata, colordata) και αρχικοποιεί τους vertex και color buffer του αντικειμένου. Επίσης, δέχεται ως ορίσματα ένα target που δηλώνει σε τι είδους buffer θα φορτωθούν τα δεδομένα (στην περίπτωση μας το target GL_ARRAY_BUFFER) και ένα όρισμα usage το οποίο δηλώνει με ποιο θα σχεδιαστούν τα τρίγωνα (στην περίπτωση μας έχουμε GL_STATIC_DRAW).

Η συνάρτηση updateObject δέχεται τα ίδια ορίσματα με την συνάρτηση InitObject και σκοπός της είναι να ενημερώνει τους buffers των αντικειμένων που έχουν υποστεί αλλαγές.

Η συνάρτηση InitDraws φορτώνει τα δεδομένα στους Shaders και χρησιμοποιεί το όρισμα loc ως signal στους Shaders για την σωστή εφαρμογή του texture και των χρωμάτων. Στην συνέχεια, δεσμεύει τον αντίστοιχο χώρο, ανάλογα με το τι αντικείμενο θέλουμε να σχεδιάσουμε.

Η συνάρτηση collition_with_sphere δέχεται ως ορίσματα 6 τιμές min και max για καθένα από τους άξονες x,y,z, τις οποίες και ενημερώνει με την ελάχιστη και μέγιστη τιμή που μπορεί να πάρει κάθε άξονας.

Η συνάρτηση navigateSphere δέχεται ως όρισμα την κεντρική σφαίρα, τον άξονα ως προς τον οποίο θέλουμε να μετακινήσουμε την σφαίρα και το πρόσημο της κατεύθυνσης ως προς την οποία θέλουμε να την μετακινήσουμε (1 για θετική κατεύθυνση και -1 για αρνητική).
Όταν το posneg = -1, μετατοπίζουμε την κεντρική σφαίρα προς τα αριστερά του αντίστοιχου άξονα, ενώ όταν posneg = 1 προς τα δεξιά, ενώ παράλληλα ελέγχουμε να μην ξεπερνάμε το πάνω όριο του κύβου όταν κινούμαστε προς τα δεξιά και το κάτω όριο όταν κινούμαστε προς τα αριστερά. Τέλος, ενημερώνουμε τους αντίστοιχους buffers με την συνάρτηση updateObject.

Η συνάρτηση navigateInside δέχεται ως ορίσματα τις κορυφές οποιουδήποτε τύπου αντικειμένου που δημιουργούνται, το διάνυσμα κατεύθυνσης του αντικειμένου, το id του τύπου του αντικειμένου και την κεντρική σφαίρα. Για κάθε τύπο αντικειμένου, κρατάμε το πλήθος των κορυφών. Δημιουργεί τις μεταβλητές x,y,z που προσδιορίζουν την κατεύθυνση του κύβου προς τον αντίστοιχο άξονα και ελέγχει εάν κάποιο σημείο του κύβου βρίσκεται εκτός του πάνω ή του κάτω ορίου. Όταν συμβαίνει αυτό, χρειάζεται να αντιστρέφουμε την κατεύθυνση του άξονα που βρίσκεται εκτός ορίων, εφόσον η γωνία ανάκλασης είναι ίδια με την γωνία πρόσκρουσης. Επίσης, γίνεται έλεγχος εάν ο κύβος συγκρούεται με την κεντρική σφαίρα μέσω της συνάρτησης collition_with_sphere και ισχύει ο ίδιος κανόνας με την σύγκρουση κύβου,δηλαδή η αλλαγή κατεύθυνσης.

Όταν η κεντρική σφαίρα κινείται προς κάποιο μικρότερο αντικείμενο το οποίο κινείται προς την κεντρική σφαίρα, το αντικείμενο “εγκλωβίζεται ” στην κεντρική σφαίρα μέχρι να την απομακρύνουμε από αυτό. Αυτό συμβαίνει επειδή όταν το αντικείμενο κινείται προς την σφαίρα και η σφαίρα προς το αντικείμενο και το αντικείμενο εισέρχεται στην σφαίρα,τότε το διάνυσμα κατεύθυνσης που απαιτείται για να βγει γίνεται ολοένα και μεγαλύτερο.

Στην main του προγράμματος, γίνεται η ενεργοποίηση των κατάλληλων βιβλιοθηκών της OpenGL, και ύστερα η δημιουργία του παραθύρου της εφαρμογής, διαστάσεων 600 x 600, με τίτλο “Συγκρουόμενα” και μαύρου χρώματος.

Στην συνέχεια, κάνουμε load το texture με την συνάρτηση stbi_load και φορτώνουμε τα δεδομένα στον buffer για το texture.

Φορτώνουμε τους Shaders με την εντολή LoadShaders.

Ύστερα, ορίζουμε τις uniform μεταβλητές MVP και InSignal οι οποίες χρησιμοποιούνται στους Shaders για επικοινωνία του προγράμματος με τους Shaders.

Μετά από τον ορισμό των uniform μεταβλητών, δημιουργούμε τον μεγάλο κύβο, την κεντρική σφαίρα και τον πίνακα με τις u,v συντεταγμένες του texture, τον οποίο και προσθέτουμε στον αντίστοιχο buffer.

Πατώντας το πλήκτρο space, δημιουργείται ένας τυχαίος αριθμός στο διάστημα [1,3]. Όπως εξηγήθηκε παραπάνω, οι αριθμοί αυτοί υποδηλώνουν τι αντικείμενο παράγεται κάθε φορά (1 για κύβο, 2 για σφαίρα, 3 για κύλινδρο) και ανάλογα με το αποτέλεσμα της rand() παίρνουμε το αντίστοιχο αντικείμενο. Για κάθε αντικείμενο, φτιάχνουμε ένα τυχαίο διάνυσμα κατεύθυνσης στο διάστημα [0.1 , 0.9] και το προσθέτουμε στο διάνυσμα που κρατά τα αντικείμενα. Μετά από αυτή τη διαδικασία, ενεργοποιούμε τους buffers των αντικειμένων.

Με το πάτημα του πλήκτρου “ > ” επιτυγχάνεται η αύξηση της ταχύτητας των αντικειμένων που κινούνται εντός του κύβου (εκτός την κεντρική σφαίρας και του μεγάλου κύβου), ενώ με το πάτημα του “ < ” η μείωση της ταχύτητας των αντικειμένων. Με παρατεταμένο πάτημα του κάθε πλήκτρου είναι πιο εμφανείς οι αλλαγές στην ταχύτητα.

Για κάθε αντικείμενο, στην συνέχεια, πραγματοποιούμε κίνηση εντός του κύβου και ενημερώνουμε κάθε φορά τους buffers.

Σχετικά με το οπτικό πεδίο της κάμερας, υπολογίζουμε τον MVP μετασχηματισμό, τον οποίο και στέλνουμε στους Shaders.

Για καθένα από τα πλήκτρα “ ^ ”, “ v ”, ” >”, “ < ”, “ + ”,” - ” εκτελείται η κάθε μετατόπιση που ζητείται στην εκφώνηση της προγραμματιστικής άσκησης.

Τα αντικείμενα, εκτός της κεντρικής σφαίρας και του μεγάλου κύβου, σχεδιάζονται με την συνάρτηση initDraws. Αφού σχεδιαστούν, σχεδιάζονται πιο κάτω η κεντρική σφαίρα και και ο μεγάλος κύβος, έτσι ώστε να επιτευχθεί το transparency του μεγάλου κύβου και το solid χρώμα των εσωτερικών αντικειμένων.

Με το πάτημα του πλήκτρου T, στέλνουμε στους Buffers τα κατάλληλα δεδομένα για το texture και ανάλογα με την τιμή της μεταβλητής btn εφαρμόζουμε το texture όταν btn =1 και solid χρώμα όταν btn = -1.

Με το πάτημα του πλήκτρου escape τερματίζει το παράθυρο και διαγράφουμε τους buffers που χρησιμοποιήθηκαν.

Αναφορικά με τον κώδικα των Shaders, το location = 0 δέχεται τις κορυφές των σχημάτων, το location = 1 τα χρώματα τους και το location = 2 τις συντεταγμένες του texture. To output fragmentColor είναι τύπου vec4 επειδή τα χρώματα που παίρνουμε ως input είναι τύπου RGBA και χρειάζεται να τα μεταβιβάσουμε ως vec4 στον fragment shader όπως και το TextCoord για την εφαρμογή του texture σε αυτές τις συντεταγμένες.
Στον ColorFragmentShader, ανάλογα με την τιμή του InSignal, αναθέτουμε το texture στην σφαίρα όταν InSignal =1 και το solid χρώμα όταν InSignal =0.

Στον κώδικα του αρχείου controls.cpp, η κάμερα κοιτάει πάντα στο κέντρο του κύβου στο σημείο (50,50,50) στο σύστημα παγκοσμίων συντεταγμένων. Αυτό μας επιτρέπει να κινούμε την κάμερα περιμετρικά του κύβου (ακολουθώντας σφαιρική τροχιά) και να κινούμαστε οριζόντια, κατακόρυφα και να πλησιάζουμε/απομακρυνόμαστε από την σκηνή.
Θέτοντας μια ακτίνα r = 2, μας επιτρέπει να ξεκινάμε σε απόσταση z=2 από το κέντρο της σφαίρας.

Με τα πλήκτρα E,X,W,S,D,A πραγματοποιούμε κίνηση της κάμερας στους άξονες x,y,z, όπως έχει εξηγηθεί παραπάνω.

Για υποστηρικτικό υλικό στην υλοποίηση της άσκησης, αξιοποιήσαμε τις ακόλουθες πηγές:

http://www.opengl-tutorial.org/beginners-tutorials/

http://www.opengl-tutorial.org/intermediate-tutorials/

http://ogldev.org/

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

http://www.opengl-tutorial.org/beginners-tutorials/tutorial-5-a-textured-cube/

https://learnopengl.com/code_viewer_gh.php?code=src/1.getting_started/4.2.textures_combined/textures_combined.cpp

https://stackoverflow.com/questions/6686741/fragment-shader-glsl-for-texture-color-and-texture-color

https://www.opengl.org/sdk/docs/tutorials/ClockworkCoders/uniform.php
https://learnopengl.com/Getting-started/Camera

https://gist.github.com/roxlu/3077861