

Portal Maze

Απαλλακτική εργασία στο μάθημα Γραφικά και Εικονική
Πραγματικότητα

Αλκίνοος Αλυσσανδράκης
1072752
5ο έτος

Πίνακας Περιεχομένων

Πίνακας Περιεχομένων.....	1
Εισαγωγή.....	2
Απαιτήσεις προγράμματος.....	3
Αυστηρές απαιτήσεις.....	3
Πρόσθετες απαιτήσεις.....	3
Υλοποίηση.....	4
Οντότητες.....	4
Entity.....	4
Camera.....	4
Light.....	5
Mesh.....	5
Material/Texture.....	5
Model.....	5
Maze.....	5
Player.....	5
Portal.....	6
Scene.....	6
Shader.....	6
Renderer.....	6
Φωτοσκίαση (Shaders).....	6
Portals.....	6
Rendering Pipeline.....	6
Εξωτερικές Βιβλιοθήκες.....	6
Βελτιώσεις.....	7
Shaders.....	7
Portals.....	7
Collisions.....	7
Εγκατάσταση & Χειρισμός.....	8
References & Resources.....	8

Εισαγωγή

Η παρούσα εργασία έγινε στο πλαίσιο του μαθήματος Γραφικά και Εικονική Πραγματικότητα της σχολής Ηλεκτρολόγων Μηχανικών και Τεχνολογίας Υπολογιστών του Πανεπιστημίου Πατρών.

Σκοπός της εργασίας είναι η δημιουργία ενός προγράμματος για υπολογιστή του τύπου video game χρησιμοποιώντας τη βιβλιοθήκη γραφικών OpenGL. Στο πρόγραμμα αυτό παρουσιάζεται στο χρήστη ένας εικονικός κόσμος που περιέχει ένα λαβύρινθο αποτελούμενο από κύβους διαφόρων υλικών και ο χρήστης αναλαμβάνει το ρόλο ενός χαρακτήρα μέσα σε αυτό τον κόσμο του οποίου ο σκοπός είναι να βρει την έξοδο και να βγει από τον λαβύρινθο.

Προκειμένου να το καταφέρει αυτό ο χρήστης μπορεί να κινηθεί εντός του λαβυρίνθου υπό τους περιορισμούς του χαρακτήρα (δεν μπορεί να περάσει μέσα από τοίχους). Έχει όμως και τη δυνατότητα να τοποθετήσει στους τοίχους δύο πύλες (portals) οι οποίες συνδέονται μεταξύ τους. Αυτές οι πύλες συνδέουν δύο σημεία στο χώρο σαν να ήταν ένα, δηλαδή κοιτώντας μέσα από τη μία πύλη φαίνεται ο κόσμος από το οπτικό πεδίο της άλλης και περνώντας μέσα από την πύλη ο χρήστης τηλεμεταφέρεται ανάμεσα στα δύο ενωμένα σημεία.

Η ιδέα των portals πρωτοεμφανίστηκε στο παιχνίδι Narbacular Drop το 2005 που δημιουργήθηκε από μια ομάδα φοιτητών του DigiPen Institute of Technology και αργότερα έγινε διάσημη με τα παιχνίδια Portal και Portal 2 της Valve που βγήκαν το 2007 και 2011 αντίστοιχα.

Απαιτήσεις προγράμματος

Αυστηρές απαιτήσεις

1. Δημιουργία ενός τυχαίου λαβύρινθου στον οποίο οι τοίχοι είναι φτιαγμένοι από κύβους. Τυχαία κάποιοι από τους κύβους, που δεν βρίσκονται στην εξωτερική μεριά του λαβύρινθου, έχουν μισό ύψος σε σχέση με τους υπόλοιπους ώστε ο χρήστης να μπορεί να δει πέρα από αυτούς αλλά να μην μπορεί να περάσει από εκεί.
2. Εμφάνιση του λαβύρινθου στην οθόνη με τη μέθοδο instancing. Κάθε κύβος του τοίχου να έχει ένα τυχαίο υλικό από μια προκαθορισμένη λίστα υλικών.
3. Εισαγωγή ενός χαρακτήρα που αναπαριστά τον παίκτη. Η κίνηση του χαρακτήρα είναι περιορισμένη, μπορεί δηλαδή να κινηθεί εμπρός, πίσω, αριστερά και δεξιά, δεν μπορεί να περάσει μέσα από τοίχους και δεν μπορεί να κάνει άλματα.
4. Εισαγωγή πηγών φωτός και φωτοσκίαση της σκηνής.
5. Αποσύνδεση της κάμερας από τον χαρακτήρα και ελεύθερη κίνηση της εντός του κόσμου. Επανασύνδεση της κάμερας με το χαρακτήρα.
6. Δημιουργία portal πάνω στον τοίχο που κοιτάει ο χαρακτήρας. Τα portals μπορούν να τοποθετηθούν στους τοίχους του λαβύρινθου οι οποίοι έχουν πλήρες ύψος και έχουν ελεύθερο χώρο μπροστά τους. Όταν ένα ζεύγος portals έχει δημιουργηθεί, κοιτώντας μέσα από το ένα φαίνεται ο κόσμος από την οπτική γωνία του άλλου. Το φαινόμενο αυτό να υλοποιηθεί χρησιμοποιώντας το stencil buffer.
7. Δυνατότητα του χαρακτήρα να περάσει μέσα από το ένα portal και να βρεθεί στη θέση του άλλου portal.

Πρόσθετες απαιτήσεις

1. Φωτοσκίαση της σκηνής χρησιμοποιώντας την τεχνική Physically Based Rendering (PBR) χρησιμοποιώντας το metallic – roughness pipeline, μαζί με normal mapping και ambient occlusion.
2. Σωστή και αποδοτική χρήση των πόρων (επεξεργαστής, μνήμη, κάρτα γραφικών) στη λειτουργία του προγράμματος.
3. Χρήση μοντέρνων λειτουργιών της βιβλιοθήκης OpenGL.
4. Εφαρμογή καλών πρακτικών αντικειμενοστρεφούς προγραμματισμού.

Υλοποίηση

Οντότητες

Entity

Διαχειρίζεται όλη την πληροφορία σχετικά με τη θέση, την περιστροφή και το μέγεθος ενός αντικειμένου εντός του κόσμου.

Έχει τις εξής ιδιότητες:

- θέση (position) που εκφράζεται ως ένα τρισδιάστατο διάνυσμα (`glm::vec3`) στο σύστημα συντεταγμένων κόσμου
- προσανατολισμός (orientation) που εκφράζεται ως ένα quaternion (`glm::quat`)
- μέγεθος (scale) που εκφράζεται ως ένα τρισδιάστατο διάνυσμα (`glm::vec3`) με τους συντελεστές που χρησιμοποιούνται για την κλιμάκωση σε κάθε διάσταση

Περιέχει μεθόδους για την αλλαγή αυτών των χαρακτηριστικών όπως `setPosition`, `translate`, `rotate`, `changeScale` κ.α., καθώς και μεθόδους για την εύρεση των τριών αξόνων του τοπικού συστήματος συντεταγμένων (`forward`, `up`, `right`).

Είναι υπεύθυνη για τη δημιουργία του τελικού `model matrix` που περιγράφει αυτούς τους μετασχηματισμούς.

Οι κλάσεις `Camera`, `Model`, `Light` και `Player` κληρονομούν από αυτή την κλάση.

Camera

Διαχειρίζεται τις πληροφορίες σχετικά με την κάμερα. Κληρονομεί από την κλάση `Entity`.

Έχει τις εξής ιδιότητες (εκτός από αυτές που κληρονομεί):

- Απόσταση κοντινού επιπέδου αποκοπής (`nearCP`)
- Απόσταση μακρινού επιπέδου αποκοπής (`farCP`)
- Γωνία οπτικού πεδίου (`fov`)
- Αναλογία διαστάσεων (`aspectRatio`)
- Μετασχηματισμός στο σ.σ. κάμερας (`viewMatrix`), πίνακας 4x4 (`glm::mat4`)
- Μετασχηματισμός στο σ.σ προβολής (`projectionMatrix`) πίνακας 4x4 (`glm::mat4`)

Ο πίνακας `viewMatrix` δημιουργείται με τη συνάρτηση `glm::lookAt` χρησιμοποιώντας τη θέση και τα διανύσματα `forward` και `up` (όπως αυτά προκύπτουν από την κλάση `Entity`).

Ο πίνακας `projectionMatrix` δημιουργείται με τη συνάρτηση `glm::perspective` χρησιμοποιώντας τις ιδιότητες `fov`, `aspectRatio`, `nearCP` και `farCP`.

Οι πίνακες `viewMatrix` και `projectionMatrix` χρησιμοποιούνται αργότερα στην κάρτα γραφικών για το μετασχηματισμό σημείων και διανυσμάτων στα κατάλληλα συστήματα συντεταγμένων για το σωστό υπολογισμό του φωτισμού της σκηνής και τη σωστή προβολή τους εν τέλει στην οθόνη.

Light

Διαχειρίζεται τις πληροφορίες σχετικά με τα φώτα της σκηνής. Κληρονομεί από την κλάση Entity

Έχει τις εξής ιδιότητες (εκτός από αυτές που κληρονομεί):

- Χρώμα του φωτός (color) που αποτελείται από τρεις συνιστώσες (για το κόκκινο, το πράσινο και το μπλε) τοποθετημένες σε ένα διάνυσμα (glm::vec3)
- Ένταση του φωτός (power)

Η κλάση αυτή χωρίζεται σε δύο υποκλάσεις PointLight και DirectionalLight από τις οποίες χρησιμοποιείται μόνο η θέση ή μόνο η κατεύθυνση του φωτός αντίστοιχα για τον υπολογισμό του φωτισμού της σκηνής.

Mesh

Περιέχει τα δεδομένα ενός πλέγματος, φροντίζει για τη σωστή φόρτωση τους στην κάρτα γραφικών και εκτελεί τις εντολές για την σχεδίαση του πλέγματος.

Έχει τις εξής ιδιότητες:

- ευρετηριασμένες κορυφές (indexedVertices), μια λίστα με τα δεδομένα κάθε κορυφής, όπου κάθε κορυφή έχει θέση (position), συντεταγμένη υν (υν), κανονικό διάνυσμα (normal), εφαπτομενικό διάνυσμα (tangent)
- δείκτες κορυφών (indices), μια λίστα με δείκτες προς τη λίστα κορυφών, οι οποίοι ανά τρεις σχηματίζουν ένα τρίγωνο του πλέγματος
- δείκτης στη δομή πίνακα κορυφών για το πλέγμα (Vertex Array Object - VAO)
- δείκτης στο buffer της κάρτας γραφικών που αποθηκεύει τα δεδομένα για τις κορυφές του πλέγματος (Vertex Buffer Object - VBO)
- δείκτης στο buffer της κάρτας γραφικών που αποθηκεύει τα index των κορυφών του πλέγματος (Element Buffer Object - EBO)

Η κλάση αυτή αναλαμβάνει:

- την ανάγνωση του αρχείου obj που περιέχει τα δεδομένα για ένα πλέγμα
- να βρει και να εξαλείψει τις διπλότυπες κορυφές
- να δημιουργήσει τα εφαπτομενικά διανύσματα που χρησιμοποιούνται στο normal mapping
- να φορτώσει όλα αυτά τα δεδομένα με κατάλληλο τρόπο στη μνήμη της κάρτας γραφικών
- να εκτελέσει την εντολή σχεδίασης όταν έρθει η ώρα να εμφανιστεί το πλέγμα στην οθόνη.

Υπολογίζει επίσης το AABB του πλέγματος δηλαδή το περιβάλλον πλαίσιο που είναι ευθυγραμμισμένο με τους άξονες και χρησιμοποιείται στον έλεγχο συγκρούσεων.

Material/Texture

Οι κλάσεις Texture και Material καθορίζουν την υφή που θα έχει ένα πλέγμα όταν έρθει η ώρα να υπολογιστεί ο φωτισμός του και να εμφανιστεί στην οθόνη.

Η κλάση Texture διαβάζει ένα αρχείο εικόνας, το φορτώνει στην μνήμη της κάρτας γραφικών και το καθιστά διαθέσιμο για χρήση από την κλάση Material.

Η κλάση Material ομαδοποιεί τα διάφορα Texture που χρειάζονται για να οριστεί πλήρως ένα υλικό στην τεχνική φωτοσκίασης Physically Based Rendering (PBR).

Τα Texture και κατ' επέκταση οι ιδιότητες μιας επιφάνειας που αποθηκεύονται εντός του Material είναι:

- το βασικό χρώμα (albedo)
- η τραχύτητα (roughness)
- η μεταλλικότητα (metallic)
- τα κανονικά διανύσματα (normal)
- το ύψος (height)
- η περιβαλλοντική απόκρυψη (ambient occlusion)

Το πως χρησιμοποιούνται αυτές οι ιδιότητες στον υπολογισμό του φωτισμού θα εξηγηθεί στην ενότητα Φωτοσκίαση (Shading)

Model

Συνδυάζει ένα Mesh, ένα Material και κληρονομεί από την κλάση Entity προκειμένου να δημιουργήσει ένα ολοκληρωμένο μοντέλο με θέση, προσανατολισμό, μέγεθος, πλέγμα και υλικό.

Υπολογίζει το τροποποιημένο AABB του πλέγματος όταν αυτό μετασχηματιστεί στο χώρο από το modelMatrix του μοντέλου, προκειμένου να χρησιμοποιηθεί στον έλεγχο συγκρούσεων.

Περιέχει επίσης τη μέθοδο getClosestBlockNormal η οποία δεδομένου ενός σημείου στο χώρο υπολογίζει ποια από τις κατακόρυφες πλευρές ενός κύβου είναι πιο κοντά του και επιστρέφει ένα κάθετο διάνυσμα που την περιγράφει. Η μέθοδος αυτή χρησιμοποιείται στον έλεγχο συγκρούσεων καθώς και στη δημιουργία των portals.

Scene

Περιέχει όλα τα αντικείμενα που τοποθετούνται εντός της σκηνής.

Έχει της ιδιότητες:

- παίκτης (player)
- λαβύρινθος (maze)
- πάτωμα (floor)
- αντικείμενα σύγκρουσης (colliders)
- κατευθυνόμενα φώτα (directionalLights)
- σημειακά φώτα (pointLights)
- πύλες (portals)

Ευθύνεται μόνο για την οργάνωση των αντικειμένων της σκηνής καθώς και τη δημιουργία των portals ανάλογα σε ποιο τοίχο του λαβύρινθου κοιτάει τώρα ο παίκτης

Maze

Δημιουργεί και διαχειρίζεται τον λαβύρινθο της σκηνής.

Η κλάση αυτή αναλαμβάνει τις εξής λειτουργίες:

- δημιουργία ενός τυχαίου λαβύρινθου με συγκεκριμένο μέγεθος, όπως αυτό καθορίζεται κατά την αρχικοποίηση του αντικειμένου, με βάση τον αλγόριθμο του Prim [8]
- δημιουργία μοντέλων για όλους τους τοίχους που τοποθετούνται στη λίστα colliders του αντικειμένου Scene για να χρησιμοποιηθούν στις συγκρούσεις
- σχεδίαση των τοίχων με τη μέθοδο instancing

Κατά τη διαδικασία του instancing αντί να σχεδιάζεται το πλέγμα μόνο μια φορά αυτό σχεδιάζεται πολλές φορές, όπου κάθε φορά που σχεδιάζεται χρησιμοποιείται διαφορετικός μετασχηματισμός έτσι ώστε να τοποθετηθεί σε διαφορετικό σημείο εντός της σκηνής. Οι τοίχοι του λαβύρινθου χωρίζονται τυχαία σε ομάδες κατά τη διάρκεια της αρχικοποίησης του αντικειμένου, όπου σε κάθε ομάδα ανατίθεται ένα διαφορετικό υλικό. Όταν ο λαβύρινθος σχεδιάζεται στην οθόνη αυτό γίνεται με μια εντολή instancing για κάθε ομάδα και άρα για κάθε υλικό. Οπότε αν ο λαβύρινθος αποτελείται από 100 τοίχους και 3 υλικά, αντί να υπάρχουν 100 εντολές φόρτωσης υλικού και σχεδίασης αυτές αντικαθίστανται από 3 εντολές φόρτωσης υλικού και 3 εντολές instanced σχεδίασης.

Player

Ο χαρακτήρας που αναπαριστά τον παίκτη εντός του παιχνιδιού. Συνδυάζει ένα Model, ένα Camera και κληρονομεί από το Entity

Έχει τις εξής ιδιότητες (εκτός από αυτές που κληρονομεί):

- ταχύτητα κίνησης χαρακτήρα (movementSpeed)
- ταχύτητα κίνησης κάμερας (mouseSpeed)
- διάνυσμα ταχύτητας (velocity)
- σύνδεση/αποσύνδεση της κάμερας (noClip)
- κάμερα (camera)
- μοντέλο (model)

Η κλάση αυτή διαχειρίζεται την κίνηση του παίκτη εντός του κόσμου με βάση τους περιορισμούς που παρουσιάστηκαν στις απαιτήσεις του προγράμματος. Κάθε φορά που πατιέται ένα κουμπί κίνησης από τον χρήστη, ενημερώνεται το διάνυσμα ταχύτητας του παίκτη και με βάση αυτό σε κάθε καρέ ενημερώνεται η θέση του παίκτη και άρα η θέση του μοντέλου και της κάμερας. Ενημερώνεται επίσης και ο προσανατολισμός της κάμερας με βάση την κίνηση του ποντικιού.

Πριν γίνει η ενημέρωση της θέσης του παίκτη γίνεται έλεγχος συγκρούσεων κατά τον οποίο μπορεί να αλλάξει το διάνυσμα ταχύτητας του παίκτη.

Σε περίπτωση που η κάμερα αποσυνδεθεί από τον παίκτη αυτή μπορεί να κινηθεί ελεύθερα στο χώρο και κυρίως δεν μετακινείται το μοντέλο του παίκτη.

Portal

Ένα από τα portals που βρίσκεται εντός της σκηνής. Κληρονομεί από το Model

Έχει τις εξής ιδιότητες:

- εικονική κάμερα (camera) που χρησιμοποιείται για την προβολή της σκηνής μέσα από το portal
- συνδεδεμένο portal (linkedPortal)

Όταν δημιουργείται μόνο το ένα portal φαίνεται το πλαίσιο του. Τη στιγμή που θα δημιουργηθεί και το δεύτερο, αυτά συνδέονται και κοιτάζοντας μέσα από το ένα φαίνεται ο κόσμος από την οπτική γωνία του άλλου.

Αν ο παίκτης περάσει μέσα από το ένα portal θα βρεθεί στη θέση του άλλου portal.

Ο τρόπος με τον οποίο γίνονται αυτά θα εξηγηθεί στην ενότητα Portals.

Shader

Η κλάση αυτή διαβάζει τους shaders (vertex shader, fragmentation shader) από ένα αρχείο, τους μεταγλωττίζει, τους ενώνει σε ένα πρόγραμμα της κάρτας γραφικών και κάνει διαθέσιμο αυτό το πρόγραμμα για χρήση κατά τη διαδικασία της σχεδίασης. Φροντίζει επίσης για την εύρεση των θέσεων στη μνήμη της κάρτας γραφικών στις οποίες θα τοποθετηθούν τα διάφορα δεδομένα που χρειάζονται για τη σχεδίαση (δεδομένα κορυφών, πίνακες μετασχηματισμού, δεδομένα φωτών, υλικά)

Renderer

Διαχειρίζεται την αρχικοποίηση ορισμένων περιοχών στη μνήμη της κάρτας γραφικών και αναλαμβάνει τη σωστή σχεδίαση της σκηνής.

Οι περιοχές μνήμης είναι τα matricesUBO και lightsUBO. Η πρώτη περιέχει τους πίνακες viewMatrix και projectionMatrix όπως προκύπτουν από την κάμερα που θα χρησιμοποιηθεί για να σχεδιαστεί η σκηνή. Η δεύτερη περιέχει τα δεδομένα για όλα τα φώτα που έχουν τοποθετηθεί στη σκηνή και θα χρησιμοποιηθούν για τον υπολογισμό του φωτισμού όλων των αντικειμένων.

Κατά τη διαδικασία της σχεδίασης της σκηνής, σχεδιάζονται πρώτα τα portals και άρα οι οπτικές μέσα από αυτά και έπειτα όλη η υπόλοιπη σκηνή.

Φωτοσκίαση (Shaders)

Στο πρόγραμμα χρησιμοποιούνται τρεις διαφορετικοί shaders:

- simpleShader αποτελείται από τα αρχεία simple.vert και simple.frag, είναι πολύ απλός και εμφανίζει το αντικείμενο με ένα σταθερό χρώμα. Χρησιμοποιείται για τον σχεδιασμό των portals στο stencil και depth buffer.
- singleShader αποτελείται από τα αρχεία single.vert και pbr.frag και εκτελεί τη διαδικασία φωτοσκίασης PBR
- instancedShader αποτελείται από τα αρχεία instanced.vert και pbr.frag, είναι ίδιο με το singleShader με μικρές προσθήκες για να μπορεί να κάνει instanced rendering που απαιτείται στο σχεδιασμό του λαβύρινθου.

Ο singleShader εκτελεί τις εξής διαδικασίες:

- Στο vertex shader:
 - ο μετασχηματίζει τα δεδομένα της κορυφής (θέση, κανονικό και εφαπτομενικό διάνυσμα) στο σ.σ κόσμου
 - ο εκτελεί ορθογωνιοποίηση του εφαπτομενικού διανύσματος κατά Gram-Schmidt και υπολογίζει τον πίνακα TBN που χρησιμοποιείται στο normal mapping
 - ο μετασχηματίζει τα δεδομένα της κορυφής (θέση, κανονικό διάνυσμα) και τα δεδομένα για τα φώτα στο σ.σ. της κάμερας
 - ο μετασχηματίζει τη θέση της κορυφής στο σ.σ. προβολής και αυτή την τιμή χρησιμοποιεί ως τελική θέση της κορυφής για να προχωρήσει η διαδικασία στο fragment shader
- Στο fragment shader:
 - ο υπολογίζει το φωτισμό του fragment μέσω της διαδικασίας pbrLighting
 - ο εφαρμόζει τη διαδικασία HDR tone mapping για την επέκταση του εύρους του φωτισμού
 - ο εφαρμόζει gamma correction για τη μετατροπή του φωτισμού από το γραμμικό χώρο φωτισμού στο χώρο sRGB που χρησιμοποιούν οι περισσότερες οθόνες

Στη διαδικασία pbrLighting γίνονται τα εξής:

- αναζητούνται από τα textures οι τιμές albedo, metallic, normal, roughness, ao για το fragment
- η τιμή normal μετατρέπεται από το Tangent Space (το σ.σ. του τριγώνου στο οποίο είναι ορισμένα τα κάθετα διανύσματα, όπου αυτά κοιτάνε κατά κύριο λόγο στο +Z) στο σ.σ. του κόσμου χρησιμοποιώντας τον πίνακα TBN που υπολογίστηκε στο vertex shader
- υπολογίζεται το διάνυσμα ματιού V, δηλαδή το διάνυσμα από το το fragment προς τον παρατηρητή
- για κάθε φως στη σκηνή υπολογίζεται το διάνυσμα φωτός L από το fragment προς την πηγή φωτός αν είναι σημειακό φως ή χρησιμοποιείται κατευθείαν η κατεύθυνση του φωτός αν είναι κατευθυντικό φως, καθώς και το ενδιάμεσο διάνυσμα $H = V + L$
- για κάθε φως υπολογίζεται η CookToranceBRDF που υπολογίζει με βάση τη θέση του fragment, τα διανύσματα normal, V, L και H και τις τιμές albedo, metallic και roughness και την τιμή radiance (χρώμα και ένταση του φωτός) το χρώμα του fragment

- το χρώμα του fragment προκύπτει από το άθροισμα των χρωμάτων από όλα τα φώτα (CookToranceBRDF) στο οποίο προστίθεται και ο περιβάλλον φωτισμός όπως προκύπτει από τις τιμές albedo και ao

Η συνάρτηση CookToranceBRDF υπολογίζει το χρώμα του fragment με τέτοιο τρόπο ώστε το αποτέλεσμα να είναι πιο κοντά στη συμπεριφορά του φωτός στον πραγματικό κόσμο (κυρίως στο κομμάτι της διατήρησης της ενέργειας) απ' ότι η διαδικασία Phong στην οποία το διάχυτο και ανακλώμενο φως ορίζονται αυθαίρετα.

Περισσότερες πληροφορίες για τη διαδικασία μπορούν να βρεθούν στις αναφορές [2] και [3]

Portals

Για τον σχεδιασμό του portal πρέπει:

- να βρεθεί η οπτική γωνία του συνδεδεμένου portal, δηλαδή η θέση της εικονικής κάμερας σε σχέση με το συνδεδεμένο portal να είναι αντίστοιχη με τη θέση του παίκτη σε σχέση με το πρώτο portal
- να τροποποιηθεί ο πίνακας μετασχηματισμού στο σ.σ. προβολής έτσι ώστε να μην φαίνονται τα αντικείμενα πίσω από το portal, αφού είναι πιθανό να υπάρχουν αντικείμενα ανάμεσα στην εικονική κάμερα και την επιφάνεια του συνδεδεμένου portal
- να δημιουργηθεί ένα πλαίσιο πάνω στον τοίχο που είναι τοποθετημένο το portal έτσι ώστε μόνο σε αυτό το πλαίσιο να σχεδιαστεί κάτι
- εντός αυτού του πλαισίου να σχεδιαστεί η σκηνή χρησιμοποιώντας την εικονική κάμερα

Για να βρούμε την οπτική γωνία του συνδεδεμένου portal παίρνουμε τους εξής μετασχηματισμούς που θα εφαρμόσουμε στη θέση και τον προσανατολισμό του παίκτη:

- μετασχηματισμός από το σ.σ. κόσμου στο τοπικό σ.σ. του portal (αντίστροφο modelMatrix του πρώτου portal)
- περιστροφή κατά 180 μοίρες (δημιουργείται από τη συνάρτηση glm::rotate)
- μετασχηματισμό από το σύστημα συντεταγμένων του άλλου portal στο σύστημα συντεταγμένων του κόσμου (ταυτίζεται με το modelMatrix του συνδεδεμένου portal)

Οι μετασχηματισμοί εφαρμόζονται με τη σειρά που αναφέρονται (στη συνάρτηση getViewFromLinkedPortal) στη θέση και τον προσανατολισμό του παίκτη και προκύπτει μια νέα θέση και ένας νέος προσανατολισμός. Η ίδια διαδικασία θα χρησιμοποιηθεί και όταν θα έρθει η ώρα να τηλεμεταφερθεί ο παίκτης όταν περάσει μέσα από το portal.

Ο μετασχηματισμός προβολής της κάμερας τροποποιείται μέσω της συνάρτησης obliqueFrustum στην οποία το κοντινό επίπεδο αποκοπής της κάμερας μετακινείται έτσι ώστε να ταυτίζεται με το επίπεδο της επιφάνειας του portal όπως αυτό φαίνεται στο σ.σ. κάμερας [7].

Το πλαίσιο στον τοίχο δημιουργείται με την σχεδίαση της επιφάνειας του portal στο stencil buffer όπου όλα τα pixel τα οποία περιέχονται εντός της επιφάνειας παίρνουν την τιμή 1 ενώ όλα τα υπόλοιπα παίρνουν την τιμή 0. Έτσι όταν στη συνέχεια σχεδιαστεί η σκηνή από την

εικονική κάμερα θα σχεδιαστούν μόνο τα pixel τα οποία έχουν την τιμή 1 στο stencil buffer ενώ όλα τα υπόλοιπα θα κρατήσουν την προηγούμενη κατάσταση τους [6].

Τέλος πριν σχεδιαστεί η υπόλοιπη σκηνή σχεδιάζεται ξανά η επιφάνεια του portal μόνο στο depth buffer έτσι ώστε να μην επικαλυφθεί η εσωτερική οπτική που σχεδιάστηκε προηγουμένως.

Στην περίπτωση της τηλεμεταφοράς το πρόγραμμα κοιτάζει την τωρινή και την προηγούμενη θέση του παίκτη. Εφόσον ο παίκτης συγκρούεται με το portal και οι δύο αυτές θέσεις βρίσκονται εκατέρωθεν της επιφάνειας του portal τότε λέμε ότι ο παίκτης πρέπει να τηλεμεταφερθεί. Οπότε με βάση την τωρινή θέση και προσανατολισμό του παίκτη που βρίσκεται πέρα από την επιφάνεια του portal υπολογίζεται τα αντίστοιχα σε σχέση με το συνδεδεμένο portal (από τη συνάρτηση getViewFromLinkedPortal) και πολύ απλά θέτουμε τη θέση και τον προσανατολισμό του παίκτη ίσο με τις καινούργιες αυτές τιμές.

Collisions

Οι συγκρούσεις στο συγκεκριμένο πρόγραμμα έχουν γίνει με εξαιρετικά απλοϊκό τρόπο. Χρησιμοποιείται το AABB των μοντέλων και έτσι όταν το AABB του παίκτη επικαλύπτεται με το AABB οποιουδήποτε μοντέλου λέμε ότι υπάρχει σύγκρουση. Σε περίπτωση που ο παίκτης συγκρούεται με ένα portal το διαχειρίζεται αυτό το ίδιο το portal όποτε και αποφασίζει αν ο παίκτης πρέπει να τηλεμεταφερθεί ή όχι. Σε περίπτωση που ο παίκτης συγκρούεται με κάποιον από τους τοίχους του λαβύρινθου τότε βρίσκεται η πιο κοντινή επιφάνεια του κύβου σε σχέση με τη θέση του παίκτη, υπολογίζεται το κανονικό διάνυσμα αυτής της επιφάνειας και στη συνέχεια στην ταχύτητα του παίκτη προστίθεται αυτό το διάνυσμα επί την ταχύτητα κίνησης. Με αυτό τον τρόπο αντί ο παίκτης να εισέλθει μέσα στον τοίχο μένει μόνο στο εξωτερικό του.

Εξωτερικές Βιβλιοθήκες

Στη συγκεκριμένη υλοποίηση χρησιμοποιούνται οι εξής εξωτερικές βιβλιοθήκες:

- GLFW
 - δημιουργία και διαχείριση του παραθύρου στο οποίο προβάλλεται το αποτέλεσμα του προγράμματος
 - λήψη και διαχείριση εισόδου από το ποντίκι και το πληκτρολόγιο
- OpenGL Extension Wrangler (GLEW)
 - συναρτήσεις για τη διαχείριση της κάρτας γραφικών, όπως αποστολή δεδομένων, αλλαγή κατάστασης και εντολές σχεδίασης.
- OpenGL Mathematics (glm)
 - τύποι δεδομένων όπως glm::vec3, glm::mat4, glm::quat οι οποίοι χρησιμεύουν στα γραφικά
 - συναρτήσεις για τη δημιουργία μετασχηματισμών κάμερας και προβολής
 - συναρτήσεις για την εύκολη μετατροπή και τροποποίηση των προηγούμενων
 - λοιπές χρήσιμες συναρτήσεις μαθηματικών
- Simple OpenGL Image Library 2 (SOIL2)
 - ανάγνωση εικόνων και απευθείας φόρτωση τους στην κάρτα γραφικών ως texture

- tinyobjloader
 - ανάγνωση αρχείων waveform (.obj) που περιέχουν δεδομένα για ένα πλέγμα, εξαγωγή των επιμέρους παραμέτρων των κορυφών όπως θέση, κανονικό διάνυσμα, συντεταγμένες υν και πληροφορίες για τις έδρες του πλέγματος

Βελτιώσεις

Το πρόγραμμα στην παρούσα του μορφή είναι λειτουργικό, αλλά χρήζει αρκετών βελτιώσεων συγκεκριμένα στις τεχνικές φωτοσκίασης, στη λειτουργία των portals και στο σύστημα συγκρούσεων. Επίσης παρόλο που το πρόγραμμα είναι σχετικά αποδοτικό δεν έχει γίνει μεγάλη προσπάθεια όσον αφορά τη βελτιστοποίηση της απόδοσης και αρκετά σημεία του κώδικα θα μπορούσαν να γραφτούν ξανά για τη μεγιστοποίηση της απόδοσης.

Shaders

Οι τεχνικές φωτοσκίασης που χρησιμοποιούνται είναι αρκετά μοντέρνες αλλά από αυτές λείπει η τεχνική του parallax mapping καθώς και κάποιο σύστημα για τον υπολογισμό των σκιών που δημιουργούνται από τα αντικείμενα εντός της σκηνής.

Portals

Τα portals λειτουργούν σε ικανοποιητικό βαθμό, αλλά παρατηρείται μια απότομη εναλλαγή στην εικόνα (flicker) κάθε φορά που ο παίκτης περνάει μέσα από αυτά. Αυτό οφείλεται στο γεγονός ότι ένα portal είναι μια δισδιάστατη επιφάνεια πάνω στην οποία προβάλλεται η οπτική του άλλου portal, οπότε καθώς ο χρήστης περνάει μέσα από αυτό υπάρχει ένα σημείο στο οποίο η κάμερα περνάει πέρα από αυτή την επιφάνεια και βλέπει τα αντικείμενα από πίσω για ακριβώς ένα καρέ πριν γίνει η τηλεμεταφορά του παίκτη στη νέα τοποθεσία όπου και βλέπει τη σωστή εικόνα. Το πρόβλημα αυτό μπορεί να φτιαχτεί με διάφορες τεχνικές, μια εκ των οποίων είναι το portal να έχει όγκο αντί να είναι δισδιάστατη επιφάνεια και άρα όταν η κάμερα βρίσκεται εντός αυτού του όγκου να φαίνεται μόνο η οπτική από το άλλο portal.

Επίσης δεν έχει υλοποιηθεί η περίπτωση όπου τα δύο portals κοιτάνε το ένα το άλλο. Σε αυτή την περίπτωση κανονικά θα έπρεπε να γίνεται αναδρομικός σχεδιασμός της σκηνής όμως στη συγκεκριμένη υλοποίηση φαίνεται απλά το πλαίσιο του portal.

Collisions

Το σύστημα των συγκρούσεων είναι αυτό που παρουσιάζει τα περισσότερα και μεγαλύτερα προβλήματα. Οι συγκρούσεις και γενικότερα το σύστημα φυσικής είναι ένα από τα πιο δύσκολα σημεία υλοποίησης για οποιοδήποτε παιχνίδι. Αυτός είναι και ο λόγος που τις περισσότερες φορές προτιμάται κάποια έτοιμη υλοποίηση όπως στις περιπτώσεις των game engines όπως Unity, Unreal Engine, Godot. Το σύστημα συγκρούσεων όπως υλοποιείται στο συγκεκριμένο πρόγραμμα είναι εξαιρετικά απλοϊκό οπότε είναι πολύ εύκολο να βρεθούν περιπτώσεις στις οποίες αυτό καταρρέει και προκύπτει μη επιθυμητή συμπεριφορά. Μερικά παραδείγματα περιλαμβάνουν:

- Όταν ο παίκτης κινείται παράλληλα με ένα τοίχο και πολύ κοντά σε αυτόν τότε σε κάθε ακμή ανάμεσα στους κύβους “κολλάει” και πρέπει να μετακινηθεί λίγο πλευρικά για να συνεχίσει
- Αν ο παίκτης συγκρουστεί με ένα τοίχο αλλά συνεχίσει να προχωράει μπροστά κουνώντας την κάμερα αριστερά-δεξιά μπορεί να περάσει μέσα από τον τοίχο στο εσωτερικό του.


Εγκατάσταση & Χειρισμός

Οι οδηγίες μεταγλώττισης και εκτέλεσης του προγράμματος βρίσκονται στο αρχείο README.md

Οδηγίες χειρισμού:

- WASD: μετακίνηση του παίκτη στον κόσμο (εμπρός, αριστερά, πίσω και δεξιά κίνηση αντίστοιχα)
- Up Down arrow: ζουμ της κάμερας (μεγαλύτερο, μικρότερο αντίστοιχα)
- Left mouse click: τοποθέτηση πρώτου portal
- Right mouse click: τοποθέτηση δεύτερου portal
- Space: ελεύθερη κίνηση της κάμερας και επαναφορά της στη θέση του παίκτη
- Esc: τερματισμός παιχνιδιού

References & Resources

1. [Learn OpenGL](#)
2. [PBR Theory](#)
3. [PBR Lighting](#)
4. [Normal Mapping](#)
5. [OpenGL Documentation](#)
6. [Rendering recursive portals with OpenGL - th0mas.nl](#)
7. [Oblique View Frustum Depth Projection and Clipping](#)
8. [OpenGL Programming/Mini-Portal](#)
9. [OpenGL Programming/Mini-Portal Smooth](#)
10. [Free PBR Materials](#)
11. [Prim Maze Generation Algorithm](#)
12.  Coding Adventure: Portals