

Απαλλακτική Εργασία Γραφικών

Djinn Mesh Manipulation

Ονοματεπώνυμο: Αλέξανδρος Τσαπάρας

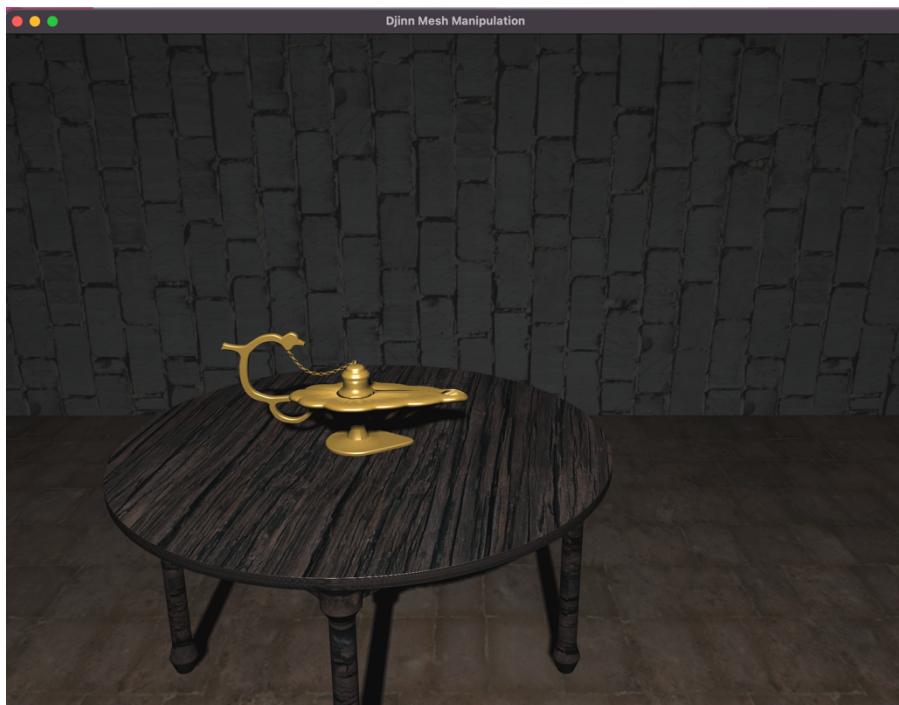
ΑΜ: 1072824

Έτος: 4ο

Περίληψη

Η εργασία μου αναπαριστά ένα λυχνάρι, μέσα από το οποίο αναδύεται ένα τζίνι. Αρχικά προβάλλω όλα τα OBJs της εργασίας, δηλαδή το δωμάτιο(το οποίο αποτελείται από 6 planes), το τραπέζι, το λυχνάρι, το τζίνι, τα σύννεφα, τα κέρματα και τα quads που χρησιμοποιώ για τα δύο particle effects. Έπειτα, για να είναι η εργασία μου πιο αληθοφανής, χρησιμοποιώ τεχνικές φωτισμού και σκίασης, τις οποίες θα αναλύσω παρακάτω. Υπάρχουν συγκεκριμένες λειτουργίες για την σωστή χρήση του προγράμματος όπως το τρέμουλο του λυχναριού (Πλήκτρο 1), το διαγώνιο zoom out/in (Πλήκτρα Z,X αντίστοιχα), για να βρεθεί ο χρήστης σε τέλεια θέση παρακολούθησης, τον καπνό και την ανάδυση του τζίνι από την μύτη του λυχναριού και επιτλέον το σταμάτημα του τρέμουλου (Πλήκτρο 2) και την βροχή από χρυσά κέρματα (Πλήκτρο 3). Τέλος συμπεριλαμβάνω και τις βασικές λειτουργίες όπως την κίνηση της κάμερας (Πλήκτρα Q, W, E, A, S, D – για κάτω, μπροστά, πάνω, αριστερά, πίσω, δεξιά αντίστοιχα), την κίνηση της λάμπας (Πλήκτρα Y, U, I, H, J, K – για κάτω, μπροστά, πάνω, αριστερά, πίσω, δεξιά αντίστοιχα) και το κλείσιμο του προγράμματος (Πλήκτρο Escape).

Μέρος Α



- Αρχικά φόρτωσα ένα μοντέλο λυχναριού, αντί του δοσμένου μοντέλου τσαγιέρας, για να φαίνεται πιο ρεαλιστικό. Μέσω του blender, αφαίρεσα το

Απαλλακτική Εργασία Γραφικών

Djinn Mesh Manipulation

πρόσθετο plane του obj αρχείου και επιπλέον το μετακίνησα κατάλληλα, ώστε η μύτη του λυχναριού να βρίσκεται στο σημείο (0,0,0). Έπειτα μου ζητήθηκε να δημιουργήσω ένα plane, για να στερεώσω το λυχνάρι, αλλά εγώ προτίμησα να φορτώσω ένα μοντέλο τραπεζιού, το οποίο παρομοίως μετακίνησα μέσω του blender, έτσι ώστε όταν το φορτώνω στο πρόγραμμα, να φαίνεται το λυχνάρι τοποθετημένο στην μέση του τραπεζιού. Τα παραπάνω τα προβάλλω μέσω των συναρτήσεων lighting_pass και depth_pass (που θα αναλυθούν αργότερα), γιατί θέλω και να φωτίζονται, αλλά και να διακρίνουμε τις σκιές τους.

- Για τον φωτισμό και την σκίαση συμβουλεύτηκα τον κώδικα του 6^{ου} εργαστηρίου και διάλεξα directional light. Αρχικά για την υλοποίηση του χρησιμοποίησα τους shaders που είχα ήδη δημιουργήσει για να κάνω render τα αντικείμενα του προηγούμενου ερωτήματος. Απλά προσθέτω νέες μεταβλητές όπως light ambient, diffuse, specular and power, για να ορίσω ποια fragments θα φωτίζονται και ποια όχι. Δημιουργησα μια συνάρτηση phong(visibility), η οποία ανάλογα με το Ka, Kd, Ks, Ns, για το υλικό του αντικειμένου και τα La, Ld, Ls, power, για τον φωτισμό του αντικειμένου, μας επιστρέφει το χρώμα του κάθε fragment. Επιπλέον χρησιμοποιώ μια συνάρτηση ShadowCalculation, η οποία υπολογίζει την σκιά του κάθε fragment, την οποία εκμεταλλεύομαι για να υπολογίσω το visibility.

Επιπροσθέτως μέσα στην phong χρησιμοποιώ δύο ακόμη μεταβλητές, την useTexture και την useTransparency.

Για την πρώτη ισχύει ότι:

- Αν είναι 1, χρησιμοποίησε τα textures του κάθε OBJ, εκτός του τζίνι και των σύννεφων.
- Αν είναι 2, textures για το τζίνι.
- Αν είναι 3, textures για το σύννεφο.

Αυτή η διαφοροποίηση γίνεται, επειδή ήθελα το τζίνι να έχει ένα πιο ανοιχτό χρώμα, το ίδιο και τα σύννεφα. Τέλος επειδή ήθελα τα σύννεφα να γίνονται πιο σκούρα όσο περνάει ο χρόνος χρησιμοποιώντας μια μεταβλητή alpha1, την οποία αυξάνω με την πάροδο του χρόνου μέσα στο do-while.

Για την δεύτερη ισχύει ότι:

- Αν είναι 1, τότε χρησιμοποίησε το alpha1 του τζίνι.
- Αν δεν έχει τιμή, τότε χρησιμοποίησε το alpha1 του κάθε αντικειμένου.

Αυτή η διαφοροποίηση γίνεται, διότι επιθυμούσα το τζίνι να ξεκινάει πολύ διάφανο από την ουρά του και όσο ανεβαίνουμε να γίνεται λιγότερο διάφανο.

Όλα τα παραπάνω διακρίνονται στην φωτογραφία του παραπάνω ερωτήματος.

Επιπλέον πρέπει να αναφέρουμε τις συναρτήσεις lighting_pass και depth_pass. Συγκεκριμένα η πρώτη επικοινωνεί με τους shaders για ShadowMapping και είναι

Απαλλακτική Εργασία Γραφικών

Djinn Mesh Manipulation

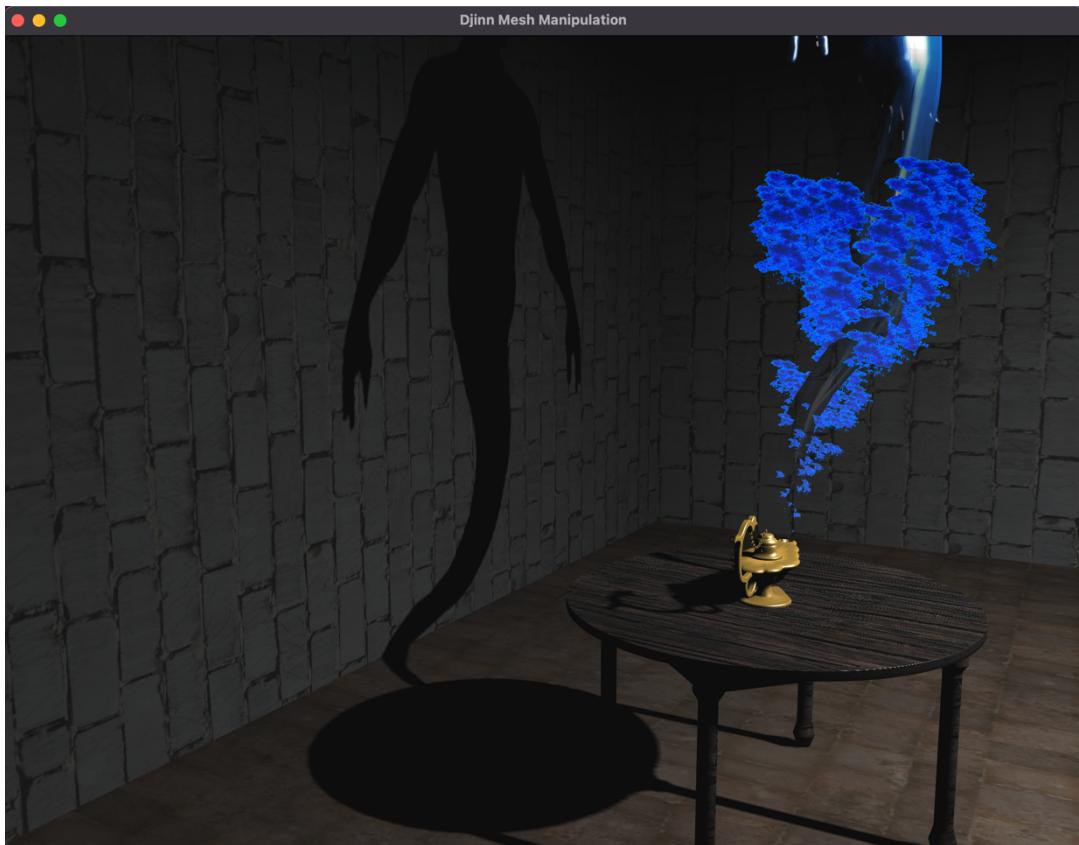
υπεύθυνη για το rendering, τον φωτισμό και την σκίαση, ενώ η δεύτερη είναι υπεύθυνη για το βάθος, δηλαδή δημιουργεί ένα depth map που χρησιμοποιείται έπειτα στο shadow mapping. Τέλος να εξηγήσω τον κάθικα που χρησιμοποίησα για να υπολογίσω το fragment color του κάθε fragment.

Ακολούθα τρία βήματα:

- **Συμβολή του ambient:** Η ambient συμβολή είναι η ίδια για όλα τα fragments στη σκηνή και δεν εξαρτάται από τη θέση της πηγής φωτός ή τη γωνία μεταξύ του φωτός και της επιφάνειας. Είναι απλώς το γινόμενο του ambient color της πηγής φωτός (`light.La`) και του ambient color του υλικού της επιφάνειας στην οποία ανήκει το fragment (`_Ka`).
- **Συμβολή του diffuse:** Η diffuse συμβολή εξαρτάται από τη γωνία μεταξύ της φωτεινής πηγής και της επιφάνειας, η οποία καθορίζεται από το εσωτερικό γινόμενο της κανονικής επιφάνειας (`N`) και το μοναδιαίο διάνυσμα που δείχνει από την επιφάνεια προς την πηγή φωτός (`L`). Εάν η γωνία μεταξύ `N` και `L` είναι μικρή (δηλαδή, η επιφάνεια είναι στραμμένη προς την πηγή φωτός), το εσωτερικό γινόμενο θα είναι μεγάλο και η diffuse συμβολή θα είναι ισχυρή. Εάν η γωνία είναι μεγάλη (δηλαδή, η επιφάνεια είναι στραμμένη μακριά από την πηγή φωτός), το εσωτερικό γινόμενο θα είναι μικρό και η συνεισφορά διάχυσης θα είναι ασθενής ή μηδενική. Το εσωτερικό γινόμενο βρίσκεται μεταξύ 0 και 1 για να αποφύγουμε αρνητικές τιμές. Οπότε η diffuse συμβολή είναι το γινόμενο του diffuse color της πηγής φωτός (`light.Ld`), του diffuse color του υλικού της επιφάνειας (`_Kd`) και του εσωτερικού γινομένου (`cosTheta`).
- **Συμβολή του specular:** Η specular συμβολή είναι η αντανάκλαση της πηγής φωτός στην επιφάνεια, και εξαρτάται από τη γωνία μεταξύ της κατεύθυνσης της ανάκλασης (`R`) και της κατεύθυνσης της κάμερας (`E`). Η κατεύθυνση της ανάκλασης υπολογίζεται χρησιμοποιώντας τη συνάρτηση `reflect`, η οποία αντανακλά το διάνυσμα -`L` (δηλαδή, την κατεύθυνση από την επιφάνεια προς την πηγή φωτός) γύρω από την κανονική επιφάνεια `N`. Η γωνία μεταξύ `E` και `R` καθορίζεται από το εσωτερικό γινόμενο των δύο διανυσμάτων, το οποίο βρίσκεται μεταξύ των τιμών 0 και 1 για να αποφύγουμε τυχόν αρνητικές τιμές. Το αποτέλεσμα υψώνεται στην δύναμη του shininess factor (`_Ns`).

Απαλλακτική Εργασία Γραφικών

Djinn Mesh Manipulation



Σκίαση

- Για την δημιουργία της καμπύλης (Bezier) που θα βγαίνει από τη μύτη του λυχναριού, αρχικά ορίζω μια νέα συνάρτηση generateCurve η οποία δέχεται σαν ορίσματα 4 σημεία (και το πλήθος των vertices που θέλουμε να μας επιστρέψει) και επιστρέφει έναν πίνακα με τα αντίστοιχα control points. Έπειτα έχω ορίσει μια συνάρτηση bezier_position, η οποία δέχεται σαν ορίσματα τα control points και επιστρέφει την θέση του κάθε particle, ξεκινώντας πάντα από την αρχική του θέση (emitter_pos). Όσον αφορά τον particle generator δημιουργώ μια κλάση IntParticleEmitter, η οποία αρχικοποιεί τα χαρακτηριστικά κάθε particle όπως position, rot_axis, rot_angle, accel(επιτάχυνση), velocity, life, mass, t (θέση του particle στην καμπύλη), p0, p1, p2, p3, control_points. Επιπλέον ορίζω δύο συναρτήσεις, την updateParticles και την createNewParticle. Η πρώτη ανανεώνει τα χαρακτηριστικά του κάθε particle όποτε «πεθάνει» και η δεύτερη δίνει τις αρχικές τιμές στα χαρακτηριστικά του κάθε particle. Επιπροσθέτως γίνονται οι απαραίτητες ενέργειες, για να μεταφέρουμε τα δεδομένα στους shaders όπως για το γνώρισμα life. Τέλος πριν αναφερθούμε στον πρώτο particle emitter, να επισημάνω δύο σημαντικές λειτουργίες (boolean μεταβλητές), την use_sorting και την use_rotations. Η πρώτη (ταξινόμηση) χρησιμοποιείται για να διασφαλίσει ότι τα particles που βρίσκονται πιο κοντά στην κάμερα να ζωγραφιστούν τελευταία, έτσι ώστε να εμφανίζονται μπροστά από particles

Απαλλακτική Εργασία Γραφικών

Djinn Mesh Manipulation

που βρίσκονται πιο μακριά. Ενώ η δεύτερη καθορίζει εάν θα εφαρμοστούν ή όχι πρόσθετα rotations στα particles με βάση τον προσανατολισμό τους. Αυτό χρησιμοποιείται συνήθως για να προσθέσει προσθέτει μεταβλητότητα στο σύστημα σωματιδίων, όπως να τα κάνει να περιστρέφονται καθώς κινούνται.

Ο πρώτος particle generator του προγράμματος μου είναι ο SmokeEmitter. Ουσιαστικά δημιουργώ 200 particles (quads) με συγκεκριμένο texture blue ink, το οποίο αποτελεσματικά φαίνεται σαν μπλε καπνός. Αρχικά για να πεθάνει ένα particle πρέπει ή να τελειώσει η ζωή του ή να ξεπεράσει ένα συγκεκριμένο ύψος (height_threshold = 5.0f). Οπότε κάθε φορά που ένα particle «πεθαίνει», δημιουργώ ένα καινούριο και έχοντας ρυθμίσει κατάλληλα την ταχύτητα και την επιτάχυνση του καθενός, δημιουργώ την συνεχή ροή που έχει ο καπνός. Επιπλέον έχω ορίσει την μεταβλητή t σαν την θέση του κάθε particle και την αλλάζω ανάλογα με τον χρόνο (dt) και την μεταβλητή counter, την οποία αυξάνω για το καθένα. Αυτό έχει ως αποτέλεσμα να ορίζω μια δεύτερη επιτάχυνση, το οποίο ευνοεί την συνεχή ροή που ανέφερα προηγουμένως. Επίσης με την πάροδο του χρόνου μειώνεται η ζωή του κάθε particle και εκμεταλλεύομενος αυτό το γεγονός αυξάνω την μάζα του καθενός, για να βελτιστοποιήσω την μορφή του καπνού. Εννοείται ότι κάνω χρήση των εξισώσεων της ταχύτητας και της επιτάχυνσης σε κάθε ανανέωση. Τελευταία ενέργεια της updateParticles είναι η συνάρτηση calculateBillboardRotationMatrix, η οποία με βοηθάει, με δεδομένα την θέση του particle και την θέση της κάμερας, να υπολογίσω τον άξονα περιστροφής, την γωνία περιστροφής και την απόσταση του καθενός από την κάμερα. Αυτό έχει ως αποτέλεσμα όλα τα particles να «κοιτάζουν» την κάμερα, όσο στρέφουμε την κάμερα. Τέλος, για να μπορέσω να δημιουργήσω ένα υποτιθέμενο πάχος του καπνού, το κάθε particle έχει την δικιά του καμπύλη bezier, οι οποίες καμπύλες ξεκινούν πάντα από το ίδιο σημείο και αυξάνονται τυχαία και στους τρεις άξονες όσο απομακρύνονται από την μύτη του λυχναριού, δημιουργώντας την κωνική μορφή που επιθυμώ.

Μπροστινή όψη του καπνού



Πλάγια όψη

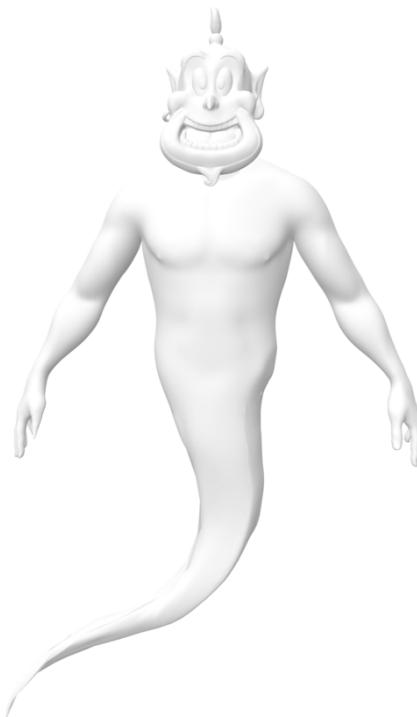


Απαλλακτική Εργασία Γραφικών

Djinn Mesh Manipulation

Μέρος Β

- Για το Djinn Mesh χρησιμοποίησα συνδυασμό δύο obj, το ένα είναι Male Mesh και το άλλο είναι μόνο το κεφάλι του τζίνι. Μέσω του blender έγινε το απαραίτητο cutting και sculpting, για να ενώσω τα objs και δημιούργια επιπλέον την ουρά του τζίνι, η οποία ξεκινάει από την μύτη του λυχναριού. Επίσης έχω συνδέσει το τζίνι με τον καπνό που ανέφερα προηγουμένως, δηλαδή όταν ενεργοποιηθεί ο particle generator του καπνού, ενεργοποιείται ταυτόχρονα το emerging του τζίνι από το λυχνάρι.



Επιπλέον πριν όλη αυτή την διαδικασία, πρέπει το λυχνάρι να τρέμει/δονείται, το οποίο το υλοποιώ μέσα στην lighting_pass κάνοντας τα εξής βήματα:

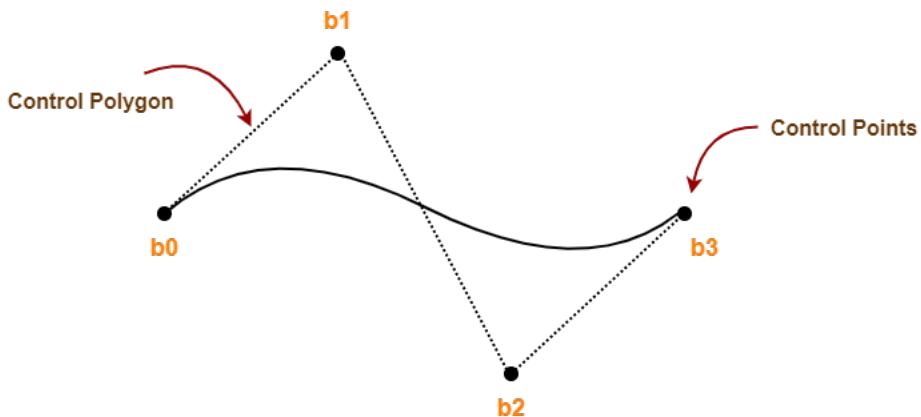
- Translation στο (0,0,0)
- Rotation στον άξονα x'x με τυχαία γωνία angleX με τιμές από -0.5f έως 0.5f
- Rotation στον άξονα z'z με τυχαία γωνία angleZ με τιμές από -0.1f έως 0.1f
- Translation στην προηγούμενη θέση του

Αυτη η διαδικασία ενεργοποιείται με το πλήκτρο 2 και συγκεκριμένα, ενεργοποιούμε μια boolean μεταβλητή tremble_action, η οποία απενεργοποιείται μόλις έχει βγει ολόκληρο το τζίνι, το οποίο ζητείται στο επόμενο ερώτημα.

- Αυτό το ερώτημα ζητάει το τζίνι να ακολουθάει την καμπύλη bezier, που ορίσαμε νωρίτερα, και ταυτόχρονα να δέχεται scaling όσο κάνει emerging από το λυχνάρι. Το υλοποιώ ακριβώς με τον ίδιο τρόπο που υλοποίησα και τον particle generator του καπνού, δηλαδή όρισα μια συνάρτηση generateBezierCurve που όπως και προηγουμένως δέχεται ως ορίσματα τα σημεία που θα ορίζουν την καμπύλη που επιθυμώ (και τον αριθμό των vertices που ζητάω ως αποτέλεσμα) και την συνάρτηση bezier_pos που δέχεται ως ορίσματα τα control_points της προηγούμενης συνάρτησης και μου επιστρέφει την θέση του τζίνι. Αρχικά δημιουργώ την μέση καμπύλη που εξέρχεται από την μύτη του λυχναριού, δηλαδή διαπερνά το κέντρο του

Απαλλακτική Εργασία Γραφικών Djinn Mesh Manipulation

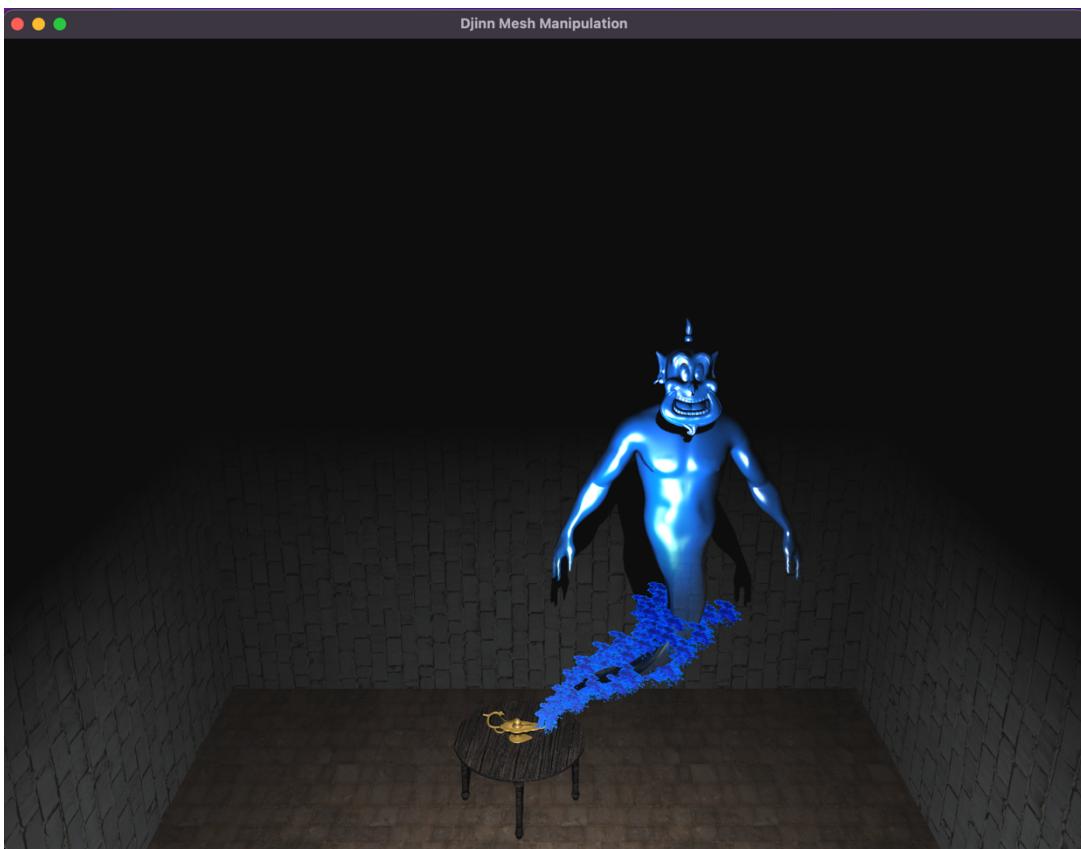
κωνικού εφέ του καπνού. Επιπλέον χρησιμοποιώ μια μεταβλητή progress, η οποία αναφέρεται στην θέση του τζίνι κάθε χρονική στιγμή, δηλαδή είναι της ίδια λογικής με το γνώρισμα των particles και με την πάροδο του χρόνου αυξάνεται εκθετικά. Έπειτα, με την χρήση της συγκεκριμένης μεταβλητής, υπολογίζω τον πίνακα που θα χρησιμοποιήσω για το translation του τζίνι, το thickness_factor με τον οποίο θα ορίσω τον πίνακα για το scaling και επίσης ανάλογα με το progress μεταβάλλω το transparency του τζίνι, δηλαδή όσο αναδύεται γίνεται λιγότερο διάφανο. Το τελευταίο το υλοποιώ με την συνάρτηση computeTransparency, η οποία παίρνει ως όρισμα το progress και επιστρέφει μια transparency τιμή. Τέλος για το τελικό transparency που θα έχει το τζίνι, συμβουλεύτηκα το 4^o εργαστήριο, δηλαδή όρισα διάφορα υψομετρικά τμήματα, τα οποία αυξάνουν με διαφορετικό ρυθμό το transparency του, το οποίο υλοποιείται στον ShadowMapping.fragmeshader όταν η μεταβλητή useTransparency είναι 1.



Συμβουλεύτηκα την παραπάνω εικόνα για να φτιάξω την Bezier καμπύλη και όπως βλέπουμε χρειαζόμαστε μόνο τέσσερα control points για να την δημιουργήσουμε.

Απαλλακτική Εργασία Γραφικών

Djinn Mesh Manipulation



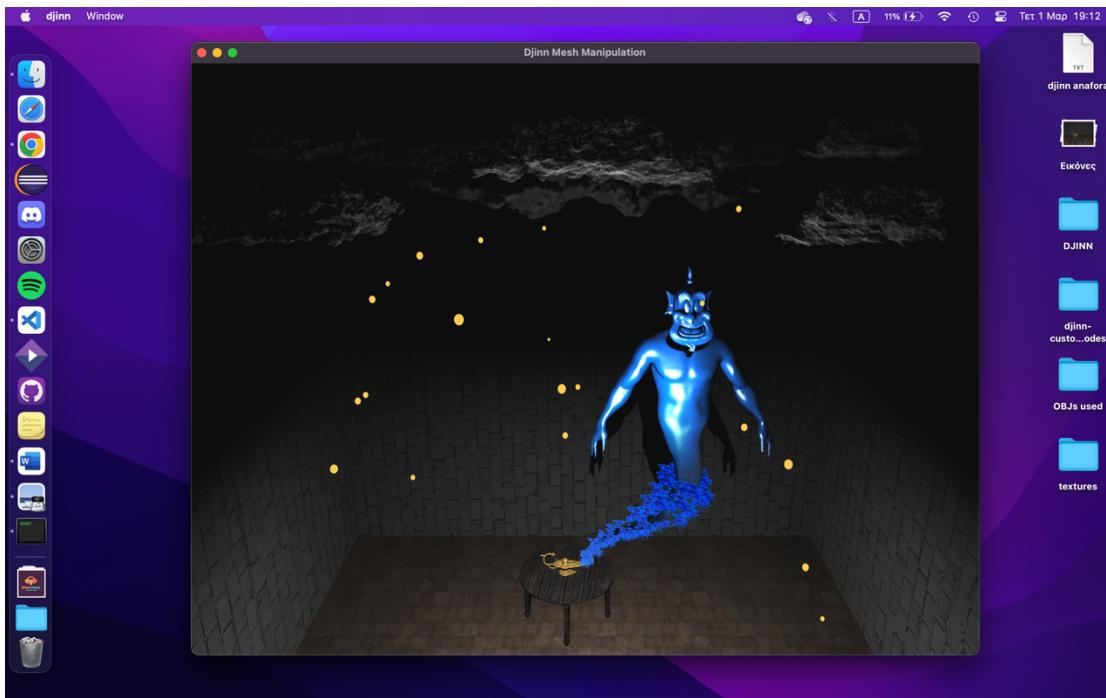
Όψη μετά το διαγώνιο zoom out

- Ο δεύτερος particle generator του προγράμματος μου είναι ο CoinRainEmitter. Ουσιαστικά δημιουργώ 200 particles (coin.obj) με συγκεκριμένο texture gold, τα οποίο αποτελεσματικά φαίνεται σαν χρυσά κέρματα. Αρχικά για να πεθάνει ένα particle πρέπει ή να τελειώσει η ζωή του ή να ξεπεράσει ένα συγκεκριμένο ύψος (height_threshold = -3.438f). Οπότε κάθε φορά που ένα particle «πεθαίνει», δημιουργώ ένα καινούριο και έχοντας ρυθμίσει κατάλληλα την ταχύτητα και την επιτάχυνση του καθενός, δημιουργώ την συνεχή ροή της βροχής από κέρματα. Εννοείται ότι κάνω χρήση των εξισώσεων της ταχύτητας και της επιτάχυνσης σε κάθε ανανέωση, παρόλο που ως επιτάχυνση έχω μόνο την βαρύτητα, έχω προσθέσει και μια ταχύτητα -10.0f στον άξονα γ'γ για να φαίνεται περισσότερο σαν χαλάζι. Επιπλέον η θέσεις των particles έχουν όλες το ίδιο ύψος (ή λίγο πιο κάτω από το emitter_pos.y), αλλά παίρνουν τυχαίες τιμές στον άξονα x'x από το πεδίο τιμών [-12.0f, 12.0f] και στον άξονα z'z από το πεδίο τιμών [-12.0f, 12.0f]. Τα rotations λειτουργούν όπως ακριβώς και στον προηγούμενο particle generator. Τέλος επηρεάζω το όριο των κερμάτων που θα βγάζει κάθε φορά, γιατί επιθυμούσα να το κάνω πιο αληθοφανές, δηλαδή να ξεκινάει με αργό ρυθμό η βροχή και με την πάροδο του χρόνου να αυξάνεται το πλήθος των κερμάτων. Αυτό το υλοποιώ πολλαπλασιάζοντας το limit με την μεταβολή του χρόνου (dt) και με έναν τυχαίο συντελεστή.

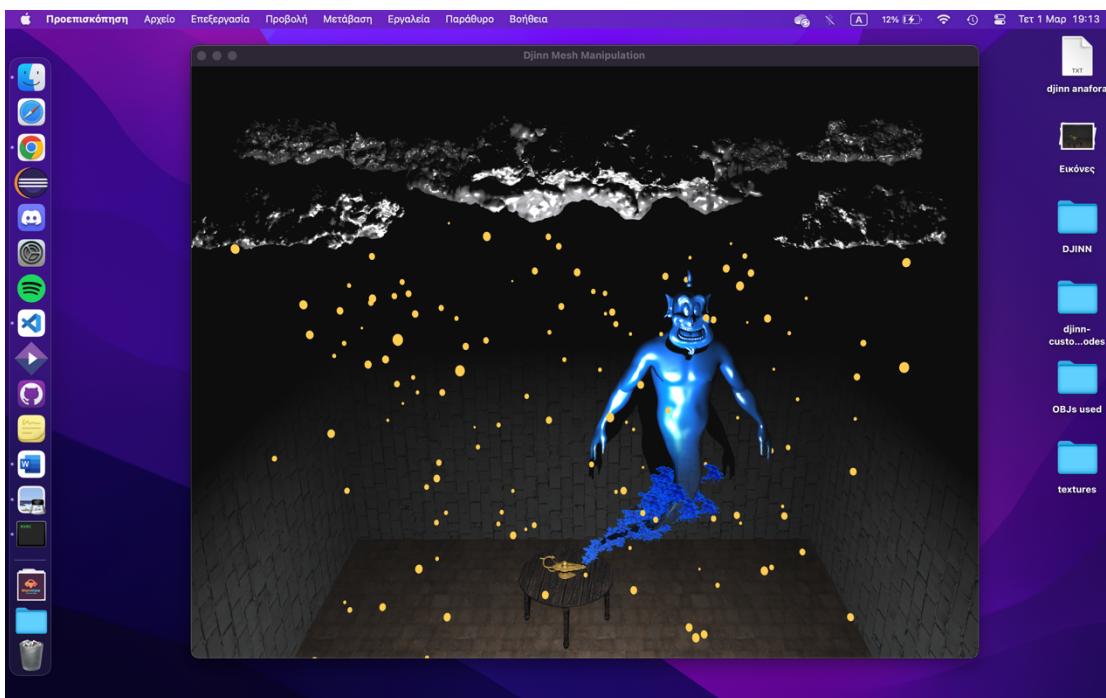
Απαλλακτική Εργασία Γραφικών

Djinn Mesh Manipulation

Για να αναπαραστήσω καλύτερα την βροχή των χρυσών κερμάτων, προσέθεσα σύννεφα (clouds.obj). Επιπλέον επηρεάζω το χρώμα και το transparency για κάθε σύννεφο με την πάροδο του χρόνου, δηλαδή όσο η μεταβλητή start_cloud_transparency είναι ενεργή, δηλαδή όποτε ξεκινάει η βροχή, τόσο μειώνουμε την διαφάνεια των σύννεφων και σκουραίνει το texture τους. Τέλος κάθε φορά που ενεργοποιείται η βροχή, η ένταση της βροχής και η διαφάνεια/ το χρώμα των σύννεφων ξεκινάει από την αρχή.



Αρχική στιγμή της βροχής χρυσών κερμάτων



Απαλλακτική Εργασία Γραφικών

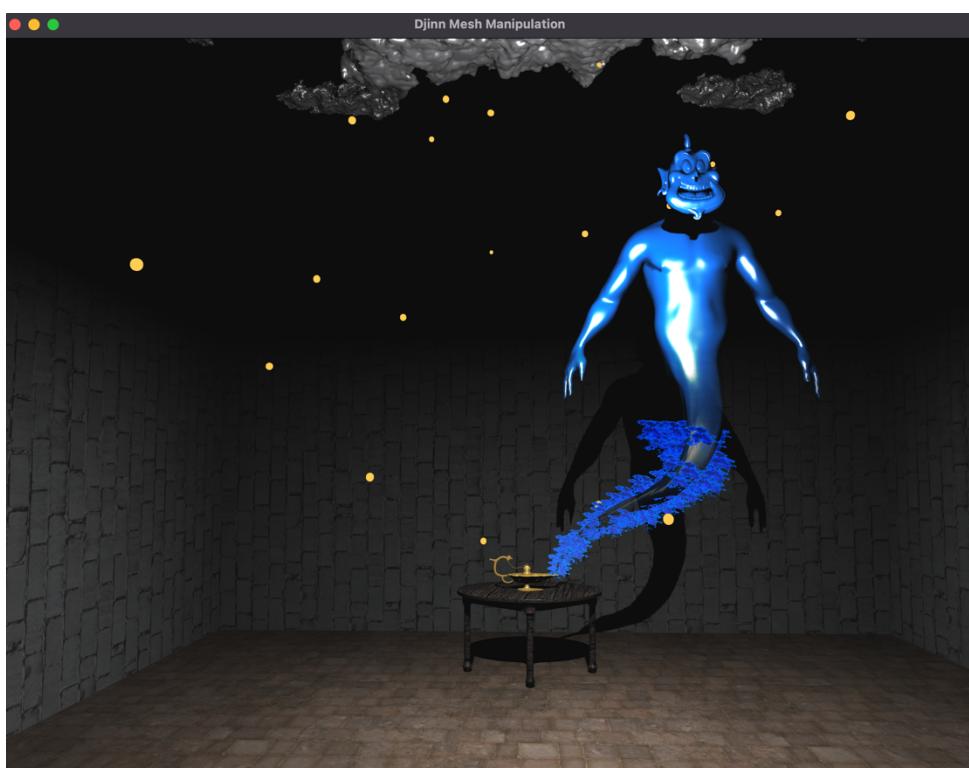
Djinn Mesh Manipulation

Τελική στιγμή της βροχής χρυσών κερμάτων

Επιπλέον Πληροφορίες

Δημιούργησα για περιβάλλον ένα δωμάτιο με πάτωμα, ταβάνι και τέσσερεις τοίχους στα οποία διακρίνονται φωτισμός και σκίαση. Επιπλέον με τα πλήκτρα Z,X κάνω zoom out/in αντίστοιχα στους άξονες γ'γ και z'z, για να παρακολουθούμε όλα τα βήματα του προγράμματος μαζί (και το οποίο υλοποιείται στο camera.cpp). Χρησιμοποιώ το πλήκτρο P για να σταματάω το πρόγραμμα, δηλαδή την ροή των particles και το πλήκτρο GLFW_KEY_GRAVE_ACCENT («`» για mac) για να ενεργοποιώ/απενεργοποιώ την χρήση του κέρσορα. Τέλος κάνω τα αντίστοιχα enables/blend για να λειτουργεί σωστά το πρόγραμμα μου:

- glEnable(GL_PROGRAM_POINT_SIZE), μου επιτρέπει να ελέγχω το μέγεθος των σημείων που σχεδιάζονται μέσω της OpenGL, ρυθμίζοντας τη μεταβλητή "gl_PointSize" στον vertex shader
- glEnable(GL_BLEND), μου δίνει την δυνατότητα να καθορίσω πώς πρέπει να συνδυάζονται τα χρώματα όταν σχεδιάζω επικαλυπτόμενα γεωμετρικά πρωτόγονα (όπως πολύγωνα) στην οθόνη.
- glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA), καθορίζει τον τρόπο με τον οποίο η τιμή alpha του εισερχόμενου pixel θα πρέπει να συνδυάζεται με τις υπάρχουσες τιμές χρώματος στον frame buffer.



Παύση προγράμματος με το πλήκτρο P