Τμήμα Ηλεκτρολόγων Μηχανικών και Τεχνολογίας Υπολογιστών Πάτρας
Τομέας Τηλεπικοινωνιών και Τεχνολογίας Πληροφορίας

Εργαστήριο Γραφικών και Εικονικής Πραγματικότητας

Ακαδημαϊκό Έτος 2020-2021

Απαλλακτική Εργασία Avalanche

ΜΑΤΑΡΑΓΚΑΣ ΜΙΛΤΙΑΔΗΣ	1046865

1. ΕΙΣΑΓΩΓΗ

Στην απαλλακτική εργασία ζητείται η δημιουργία μιας χιονοστιβάδας με την χρήση της OpenGl. Παρακάτω παρουσιάζεται η διαδικασία που ακολουθήθηκε καθώς και οι μέθοδοι εκτέλεσης των επιμέρους ερωτημάτων.

2. ΜΟΝΤΕΛΟ TERRAIN

Ως κατάλληλο μοντέλο επιλέχθηκε ένα βουνό που παρουσιάζει ανομοιογένεια στην επιφάνειά του. Το βουνό ντύθηκε με ένα texture και ακολουθήθηκε η τυποποιημένη βασική διαδικασία του pipeline της OpenGl ώστε το .obj αντικείμενο να φορτωθεί και να σταλεί στους shaders προκειμένου να ζωγραφιστεί στην οθόνη. Επιπλέον δημιουργείται και η κάμερα που επιτρέπει την κίνηση της με την χρήση των "ASWD" κουμπιών και την αλλαγή της κατεύθυνσης της με την κίνηση του κέρσορα του ποντικιού. Δεν θα επεκταθούμε στην διαδικασία υλοποίησης των παραπάνω καθώς έχουν εξηγηθεί διεξοδικά στο εργαστήριο.

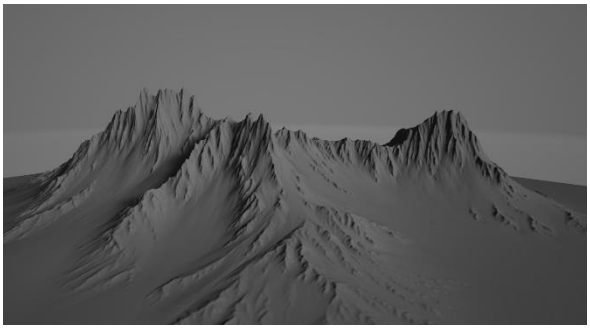

Figure 2: Mountain.obj σε λογισμικό 3D Viewer

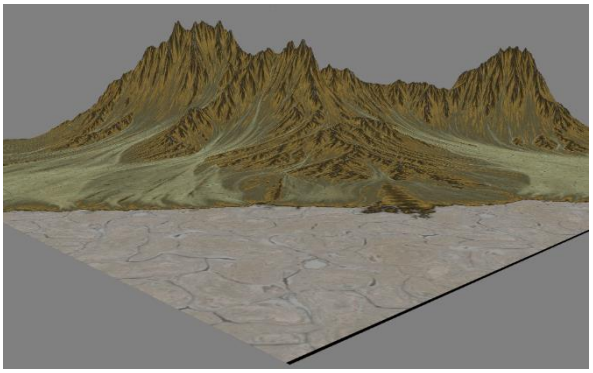

Figure 3: Το αντικείμενο όπως εμφανίζεται στο παράθυρο της OpenGl με την εφαρμογή texture

3. ΧΙΟΝΟΝΙΦΑΔΕΣ – PARTICLES

Γίνεται γρήγορα σαφές πως για την υλοποίηση της χιονόπτωσης, είναι απαραίτητη η χρήση ενός συστήματος από particles. Κάθε χιονονιφάδα πρόκειται για ένα quad δηλαδή ένα στοιχείο δύο διαστάσεων, που αποτελείτε από δύο τρίγωνα. Σε κάθε τέτοιο quad αντιστοιχεί και ένα texture χιονονιφάδας που είναι κοινό για όλα.

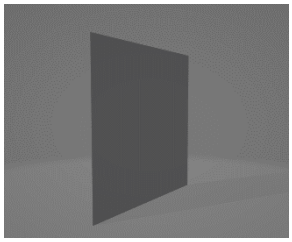

Figure 1: Quad σε 3D Viewer

Figure 4: Snowflake Texture

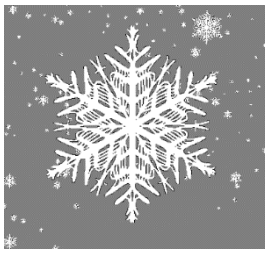

Figure 5: Quad με Texture στο παράθυρο της OpenGl

Για να ζωγραφίσουμε έναν πολύ μεγάλο αριθμό από particles ακολουθούμε την τεχνική Instancing. Ουσιαστικά, έχουμε ένα κοινό mesh, το quad μας, και δημιουργούμε διαφορετικά instances αυτού, ανάλογα με την θέση του κέντρου του. Οπότε αντί να ανανεώνουμε ένα μεγάλο mesh, με τις ακριβείς συντεταγμένες των vertices του Quad απλά χρειάζεται να τροφοδοτήσουμε έναν buffer με τις συντεταγμένες του κέντρου του.

Δημιουργούμε τα particles με τέτοιο τρόπο ώστε πάντα το normal vector τους να κοιτά την κάμερα. Η διαδικασία αυτή ονομάζεται billboarding και τα particles billboards. Πρακτικά κάνουμε rotate τα σωματίδια σε έναν συγκεκριμένο άξονα που προκύπτει από το εσωτερικό γινόμενο του διανύσματος $\vec{a}$ με το μοναδιαίο διάνυσμα $\vec{z}$, όπου

$$\vec{a} = Camera_{Position} - Particle_{Position}$$

με $a.y = 0$

Η γωνία περιστροφής προκύπτει ως:

$$\theta = \arccos(\vec{a} * \vec{z}).$$

Τα particles γεννιούνται από ένα αντικείμενο που ονομάζεται Emitter. Ο εν λόγω εκπομπός είναι υπεύθυνος για την δημιουργία των particles όπως και για τον θάνατό τους, την αρχικοποίηση των γνωρισμάτων του κάθε ενός και την ανανέωση τους ανάλογα τον χρόνο που έχει περάσει και την εξέλιξη της σκηνής. Είναι επίσης υπεύθυνος για το Instancing και το Billboarding, τα Collisions καθώς και για την δημιουργία και αποθήκευση των τριγώνων που υποδεικνύει ο αλγόριθμος Marching Cubes που θα αναλυθεί στην συνέχεια.

Όσο αφορά τα γνωρίσματα της κάθε χιονονιφάδας, δίνονται τυχαίες αρχικές επιταχύνσεις στους άξονες x και z. Στον y τους δίνεται αρνητική επιτάχυνση, αποτέλεσμα της δύναμης της βαρύτητας που τους ασκείται καθώς επίσης προστίθεται ένας όρος επιβράδυνσης ανάλογος της ταχύτητας.

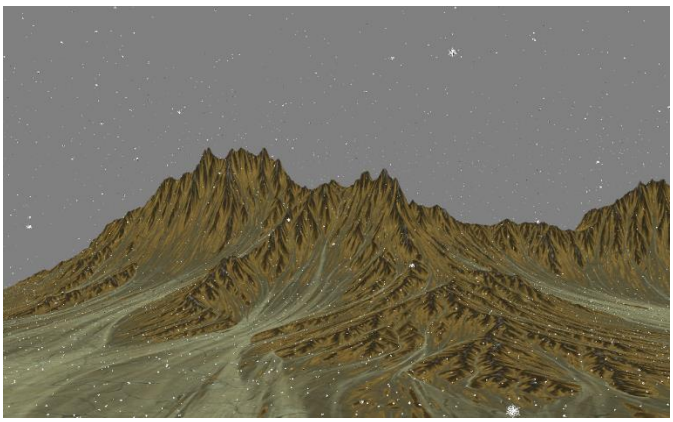

Figure 6: Χιονόπτωση με χρήση Particles

4. COLLISION DETECTION ME TERRAIN

Σε επόμενο στάδιο, το χιόνι κάθετε και να το “στρώνει” στην σκηνή. Έτσι το πρώτο βήμα είναι η αναγνώριση της σύγκρουσης με το terrain. Αυτή η διαδικασία γίνεται με την χρήση ενός “height map” που δημιουργήθηκε με την βοήθεια του λογισμικού Blender. Πρόκειται για αρχείο τύπου εικόνας που ανάλογα με το ύψος του αντικειμένου διατηρεί αποθηκευμένο αντίστοιχο χρώμα RGB για κάθε pixel. Συνεπώς σε κάθε frame και για κάθε particle γίνεται σύγκριση του ύψους του, με το ύψος του terrain που διαβάζεται από το “height map” και όταν αυτά γίνουν ίσα ή το particle πέσει κάτω από την επιφάνεια, τότε θεωρούμε πως έγινε collision.

Το Collision Handling στην προκειμένη περίπτωση έγκειται στο να ακινητοποιήσουμε τα particles στο σημείο που επιβεβαιώθηκε η σύγκρουση.

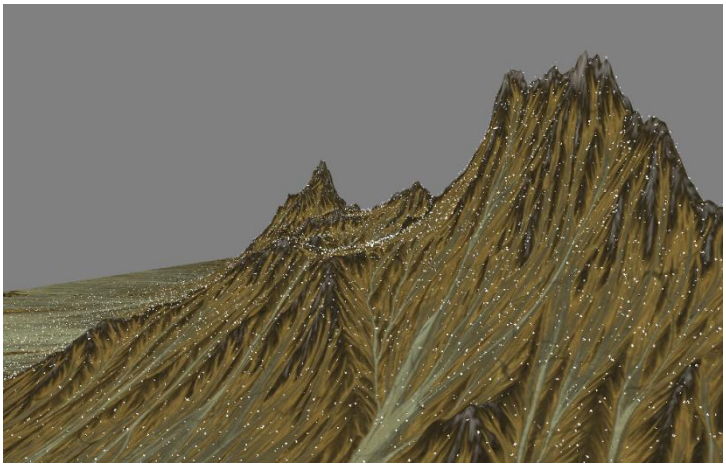

Figure 7: Τα Particles μετά την Collision Detection

5. MARCHING CUBES IMPLEMENTATION

Ωστόσο μέχρι στιγμής, το χιόνι δεν παρουσιάζεται ως στρωμένο, αλλά ως μεμονωμένες νιφάδες. Συνεπώς θα γίνει χρήση του αλγόριθμου Marching Cubes. Η βασική ιδέα είναι η αναπαράσταση ενός 3D βαθμωτού πεδίου με faces, αυτά τα faces όταν ζωγραφιστούν θα σχηματίζουν το περίγραμμα της δεδομένης γεωμετρίας.
Έτσι ο χώρος χωρίζεται σε κύβους, δημιουργείται δηλαδή ένα grid από κύβους και για κάθε έναν από αυτούς, εκτελείτε ένας έλεγχος που καθορίζει το πώς η επιφάνεια “κόβει” τον κύβο. Στην περίπτωσή μας, ο έλεγχος γίνεται για το κατά πόσο έχει γίνει Collision Χιονονιφάδας-Terrain στους 8 γείτονες του κάθε Cube και ανάλογα το αποτέλεσμα ανατίθεται η τιμή “0” ή “1”. Κάθε γείτονας αντιστοιχεί σε μια κορυφή του κύβου. Συνεπώς εξάγεται το index του κύβου που βρίσκεται υπό εξέταση.

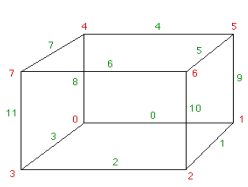

Figure 8: Κύβος με Edge και Vertex Indexes

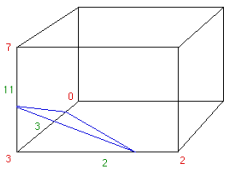

Figure 9: Παράδειγμα τριγώνου που δημιουργείται

Στην συνέχεια ο αλγόριθμος ανατρέχει στο Look Up Table που διαθέτει και αναγνωρίζει τα τρίγωνα που πρέπει να δημιουργήσει. Τα τρίγωνα αυτά με την σειρά τους εισέρχονται σε κάποιο vector και τελικά ζωγραφίζονται στην οθόνη.

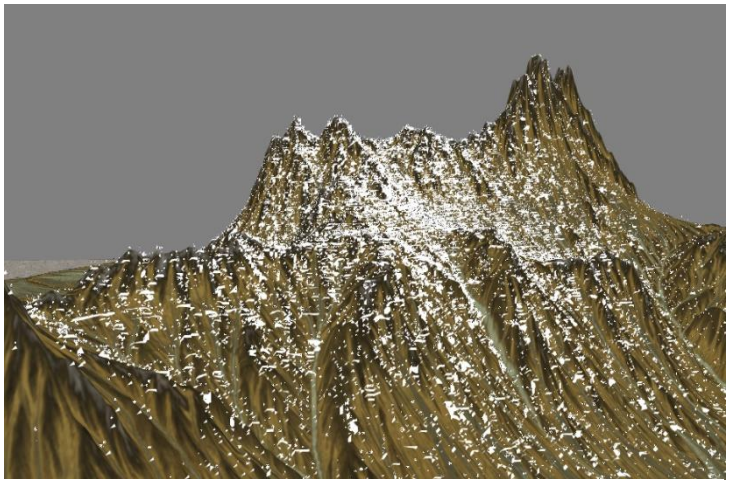

Figure 10: Στρώσιμο Χιονιού με Marching Cubes

Όπως είναι φανερό, υπάρχει σαφής βελτίωση του αποτελέσματος που λαμβάνουμε σε σχέση με τις μεμονωμένες νιφάδες. Σημαντικό είναι εδώ να σημειωθούν κάποια προβλήματα που δημιουργούνται. Αρχικά όσο μεγαλώνει ο αριθμός των particles τόσο μειώνεται και η ταχύτητα ανανέωσης των frames. Το ίδιο συμβαίνει και όσο μεγαλώνει ο αριθμός των τριγώνων που δημιουργεί ο αλγόριθμος Marching Cubes. Ο αλγόριθμος για κάθε κύβο, θα έπρεπε να ανανεώνεται κάθε φορά που πέφτει μια νιφάδα σε έναν από τους γείτονές του, όμως κάτι τέτοιο είχε πάλι ως αποτέλεσμα την ραγδαία πτώση των fps. Έτσι παρατηρούνται ασυνέχειες μεταξύ των τριγώνων.

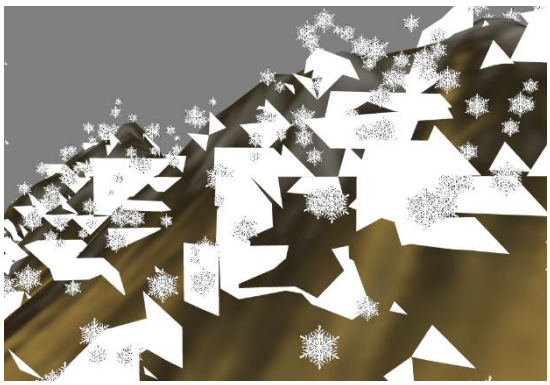

Figure 11: Ασυνέχεια στα τρίγωνα

6. RAY CASTING & ΧΙΟΝΟΣΤΙΒΑΔΑ

Στο επόμενο κομμάτι ο χρήστης μπορεί να πατήσει με τον κέρσορα του ποντικιού και ώστε να ξεκινήσει η χιονοστιβάδα. Η μέθοδος που χρησιμοποιήθηκε ώστε ο χρήσης να επιλέξει τα particles που θα ξεκινήσουν να πέφτουν είναι να προβληθεί μια τρισδιάστατη ακτίνα στην σκηνή με αρχικό σημείο την θέση της κάμερας, με το πάτημα του ποντικιού. Με κατάλληλη αναδρομική λογική στο Transformation pipeline της OpenGL μετατρέπεται το Viewport Space σε World Space.

Συνεπώς γίνεται ο έλεγχος για το αν αυτή η ακτίνα διασταυρώνεται με κάποιο particle και θεωρούμε πως αυτό το particle ξεκινά να κινείται. Ο αλγόριθμος αντιλαμβάνεται το κάθε στοιχείο που μπορούμε να διαλέξουμε ως μια σφαίρα, με κέντρο την τοποθεσία κάθε particle και με ακτίνα r, έτσι αν ο χρήστης πατήσει σε ένα σημείο, θα επιλέξει όλα εκείνα τα particles που απέχουν κάθετη απόσταση r από την ευθεία. Η σφαίρα χαρακτηρίζεται από την εξίσωση $\|P - C\| - r = 0$ όπου P οποιοδήποτε σημείο στην επιφάνεια του κύκλου και C το κέντρο της σφαίρας. Αν αντικαταστήσω το P με την εξίσωση της ευθείας δηλαδή με $\vec{R_t} = \vec{O} + \vec{D}t$ προκύπτει η δευτεροβάθμια $at^2 + bt + c = 0$ με $a = dot(D, D),\ b = 2 \cdot dot(D, O - C),$ $c = dot(O - C, O - C) - r^2$. Αν η δευτεροβάθμια δεν έχει αρνητική διακρίνουσα τότε υπάρχει τουλάχιστον ένα σημείο τομής μεταξύ της ευθείας και της σφαίρας.

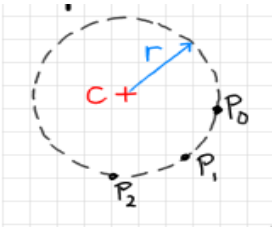

Figure 12: Σφαίρα γύρω από το Particle

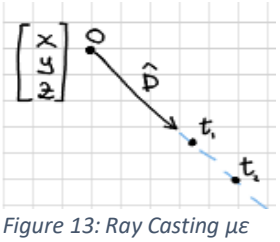

Figure 13: Ray Casting με σημείο Ο την κάμερα

Όσο αφορά την χιονοστιβάδα η λογική που εφαρμόστηκε είναι να επιτραπεί η κίνηση των particles με επιτάχυνση της βαρύτητας και δύναμη $\vec{N}$ στην κατεύθυνση του normal του τριγώνου του βουνού στο οποίο βρίσκεται το

κάθε particle. //Μέχρι στιγμής δεν γίνεται
σωστός υπολογισμός του Normal