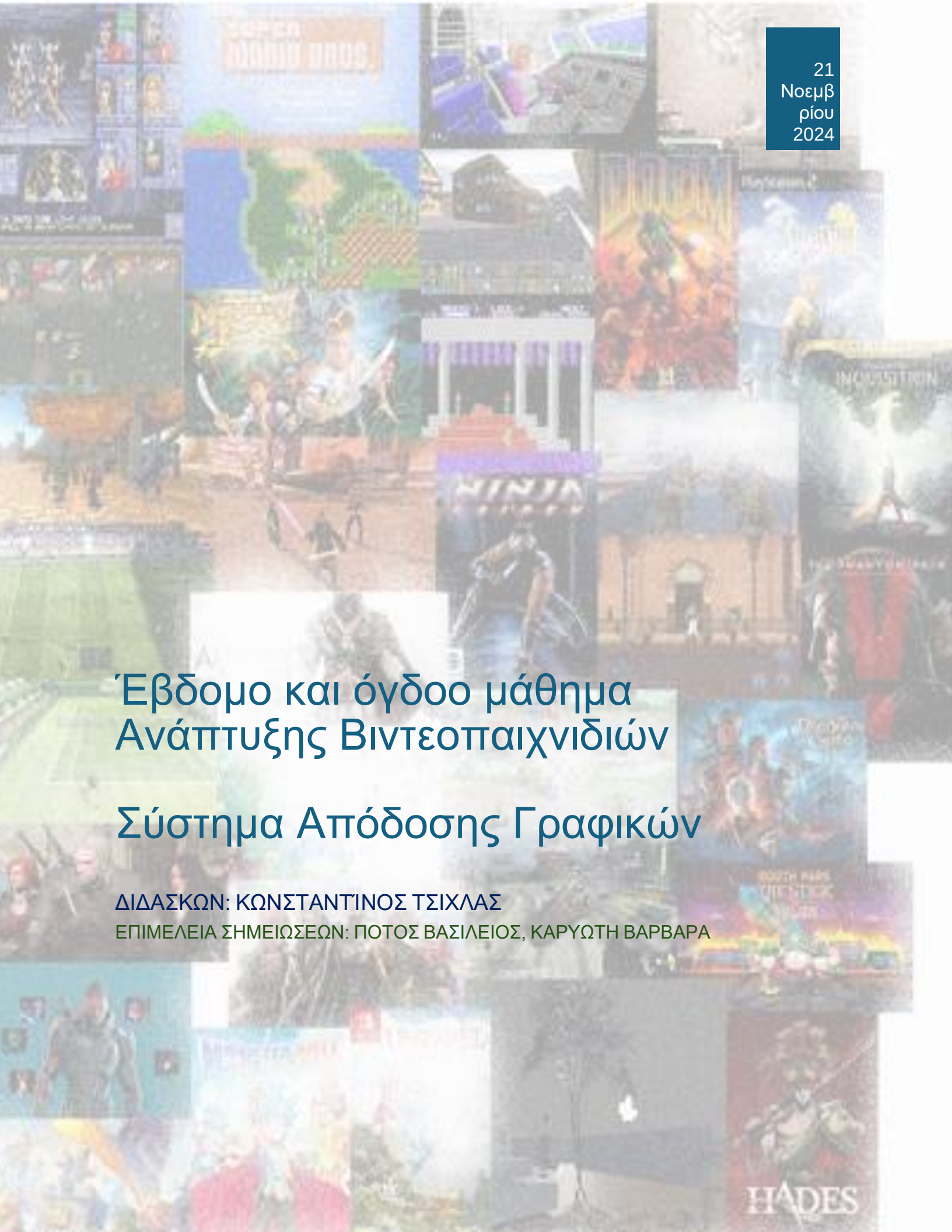


21 Νοεμβρίου 2024

Έβδομο και όγδοο μάθημα Ανάπτυξης Βιντεοπαιχνιδιών

Σύστημα Απόδοσης Γραφικών

ΔΙΔΑΣΚΩΝ: ΚΩΝΣΤΑΝΤΊΝΟΣ ΤΣΙΧΛΑΣ

ΕΠΙΜΕΛΕΙΑ ΣΗΜΕΙΩΣΕΩΝ: ΠΟΤΟΣ ΒΑΣΙΛΕΙΟΣ, ΚΑΡΥΩΤΗ ΒΑΡΒΑΡΑ

Περιεχόμενα

Σύστημα Απόδοσης Γραφικών

Εισαγωγή

Παρακάτω θα ασχοληθούμε με το σύστημα απόδοσης γραφικών εστιάζοντας στις τρεις διαστάσεις χωρίς να σημαίνει ότι πολλές από αυτές τις τεχνικές δεν εφαρμόζονται σε παιχνίδια δύο διαστάσεων. Η απόδοση 3D σκηνών περιλαμβάνει τη φόρτωση μοντέλων, τον καθορισμό κάμερας και φωτισμού, και τον υπολογισμό φωτός και χρωμάτων σε κάθε pixel για την τελική εικόνα. Θα δούμε συνοπτικά τι συμβαίνει σε διάφορα επίπεδα της επεξεργασίας για την απόδοση των γραφικών. Σε αυτά τα πλαίσια γίνεται ένας εννοιολογικός διαχωρισμός σε τρεις φάσεις, τη φάση της εφαρμογής, τη φάση της γεωμετρίας και τη φάση της ραστεροποίησης.

Αναφορά στα Voxel Engines

Στο μάθημα αυτό θα ασχοληθούμε κυρίως με μηχανές ανάπτυξης παιχνιδιών τριών διαστάσεων με πολυγωνική μοντελοποίηση των επιφανειών. Δηλαδή, προβάλουμε στην οθόνη μοντέλα τα οποία αποτελούνται από πολύγωνα (συνήθως τρίγωνα) τα οποία μπορούν να βρίσκονται οπουδήποτε στον χώρο. Σε μία μηχανή voxel, ο χώρος δεν περιγράφεται από απλά πολύγωνα, δηλαδή επιφάνειες, αλλά από voxels, δηλαδή όγκους. Μπορούμε να φανταστούμε ένα voxel σαν ένα pixel στις τρεις διαστάσεις, δηλαδή σαν ένα κύβο. Τα voxels τοποθετούνται σε ένα πλέγμα (grid) ώστε το κάθε voxel να τοποθετείται δίπλα στο επόμενο μοιράζοντας τέσσερις κορυφές. Φυσικά, όπως τα pixels έχουν όλα το ίδιο μέγεθος, έτσι και τα voxels έχουν τις ίδιες κανονικοποιημένες διαστάσεις. Οι κύριοι λόγοι να χρησιμοποιηθεί μία τέτοια μηχανή είναι γιατί διευκολύνει την δημιουργία διαδικαστικά δημιουργούμενων κόσμων, όπως στο Minecraft, καθώς και την ανάπτυξη ενός συστήματος καταστροφής περιβάλλοντος όπως στο Teardown. Τα μειονεκτήματα είναι ότι το παιχνίδι μας δεν θα φαίνεται τόσο ρεαλιστικό, αν και αυτό μπορεί να μην είναι πρόβλημα αν έχουμε αρκετά ισχυρή καλλιτεχνική κατεύθυνση, και επιπλέον αυτά τα παιχνίδια απαιτούν περισσότερη μνήμη.

Απόδοση 3D Σκηνής

Θυμίζουμε τα βασικά βήματα που απαιτούνται για την απόδοση μίας τρισδιάστατης σκηνής. Πρώτο βήμα είναι να περιγράψουμε την σκηνή μας, δηλαδή την αναπαράσταση των 3D επιφανειών. Στην πραγματικότητα, φορτώνουμε τις κορυφές και τις ακμές που αποτελούν τα πλέγματα του κάθε μοντέλου. Στη συνέχεια, τοποθετούμε την εικονική κάμερα (αναλύσαμε αυτό το βήμα σε προηγούμενη διάλεξη) και έπειτα ορίζουμε τις πηγές φωτός. Θυμίζουμε ότι το rendering είναι στην πραγματικότητα η προβολή της εφαρμογής του φωτός σε διάφορα αντικείμενα. Μετά ορίζουμε τις οπτικές ιδιότητες των επιφανειών. Το φως μπορεί να αλληλεπιδρά διαφορετικά με την κάθε επιφάνεια. Για παράδειγμα, μπορεί σε κάποια επιφάνεια να ανακλάται πλήρως, σε μία άλλη καθόλου, ενώ σε κάποια άλλη να διαχέεται σε διαφορετικές κατευθύνσεις. Τέλος υπολογίζουμε το χρώμα και το φως σε κάθε pixel όπως αυτό φαίνεται από την κάμερα.

Ροή Εργασίας για Απόδοση Γραφικών

Έχοντας κατανοήσει πως αποδίδεται θεωρητικά μία 3D σκηνή, ας εξετάσουμε τι γίνεται σε επίπεδο υλικού. Παλαιότερα κάθε βήμα θα γινόταν στον επεξεργαστή και για αυτό οι 3D σκηνές ήταν δύσκολο να αποδοθούν γρήγορα. Σήμερα όμως την περισσότερη δουλειά την κάνει η κάρτα γραφικών, η οποία χρησιμοποιεί τους χιλιάδες πυρήνες της για να παραλληλοποιήσει το κάθε βήμα, επεξεργάζοντας πολλά πολύγωνα ταυτόχρονα. Η ροή εργασίας αποτελείται από τρία εννοιολογικά στάδια: Την *εφαρμογή*, δηλαδή όλη τη φυσική και σχετικοί υπολογισμοί, τη *γεωμετρία*, δηλαδή την αναπαράσταση των μοντέλων και του χώρου, και τον ραστεροποιητή (*rasterizer*), δηλαδή την προβολή του τρισδιάστατου χώρου στην επίπεδη οθόνη μας. Το επίπεδο της εφαρμογής συνήθως σχετίζεται με τον επεξεργαστή, ενώ τα άλλα δύο επίπεδα συνήθως σχετίζονται με τη κάρτα γραφικών, χωρίς αυτό να είναι απόλυτο βεβαίως. Ας δούμε το κάθε στάδιο με λίγη παραπάνω λεπτομέρεια.

Ανάλυση σταδίων

- **Στάδιο Εφαρμογής**: Το στάδιο αυτό εκτελείται συνήθως στον επεξεργαστή και είναι άρρηκτα συνδεδεμένο με την εν λόγω εφαρμογή. Περιλαμβάνει μεταξύ άλλων υπολογισμούς φυσικής, όπως η ανίχνευση σύγκρουσης, καθώς και το υποσύστημα

τεχνητής νοημοσύνης του παιχνιδιού[1]. Ο επεξεργαστής είναι επίσης υπεύθυνος για αυτοματοποιημένες εργασίες επιπέδου μηχανής ανάπτυξης, όπως το σύστημα που παίζει τα animation (όχι κάποιο σύστημα που επιλέγει ποιο animation θα παίξει, αυτό το υλοποιεί ο προγραμματιστής, αλλά το σύστημα που είναι υπεύθυνο να παίξει το κάθε animation σωστά). Μία σημαντική επίσης εργασία είναι η μεταφορά των απαραίτητων δεδομένων στην κάρτα γραφικών μέσω του PCI Express.

- **Στάδιο Γεωμετρίας**: Όταν η κάρτα γραφικών λάβει όλα τα δεδομένα που χρειάζεται, όπως πλέγματα, πολύγωνα, υφές, θέσεις αντικειμένων, και άλλα, συνεχίζει στο στάδιο γεωμετρίας. Σε αυτό το στάδιο η κάρτα γραφικών επεξεργάζεται τα δεδομένα της ώστε να αποδοθούν στον χώρο όπως επιθυμούν οι προγραμματιστές. Πράξεις με μητρώα μετασχηματισμού για την μετακίνηση ή παραμόρφωση αντικειμένων ή της κάμερας εκτελούνται από την κάρτα γραφικών. Σε αυτό το στάδιο επίσης υπολογίζεται ο φωτισμός στις κορυφές των τριγώνων. Όταν ο χώρος είναι έτοιμος αποκόπτονται τα τρίγωνα εκτός του χώρου θέασης και η έξοδος του σταδίου γεωμετρία περνάει στο στάδιο ραστεροποίησης.

- **Στάδιο Ραστεροποίησης**: Ο ραστεροποιητής είναι υπεύθυνος για την προβολή του τρισδιάστατου χώρου στην οθόνη. Σε αυτό το στάδιο η κάρτα γραφικών θα προσθέσει και υφές και textures, καθώς και θα εκτελέσει άλλες πράξεις σε επίπεδο pixels. Μεταξύ άλλων, σε αυτό το σημείο ταξινομούνται τα τρίγωνα κατά βάθος ώστε να καθοριστεί η ορατότητα των αντικειμένων. Γενικά, επειδή αυτό το στάδιο αφορά την τελική απεικόνιση στα εικονοστοιχεία, θεωρείται εξαιρετικά αποδοτικό να γίνονται όσες περισσότερες εργασίες είναι δυνατό.

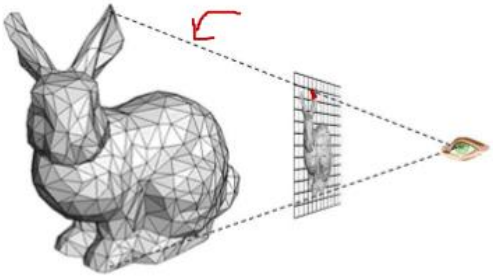
Προσεγγίσεις Απόδοσης Γραφικών

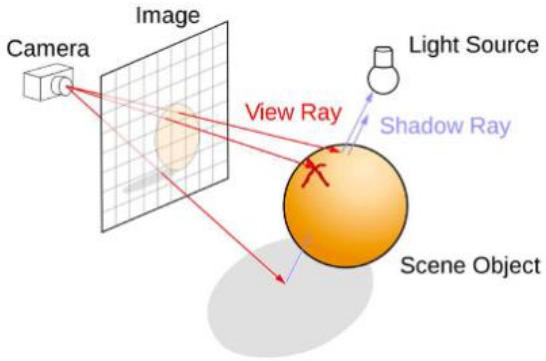
Όπως πιθανόν να γνωρίζετε, η προσέγγιση της ραστεροποίησης[2] δεν είναι η μόνη τεχνική προβολής ενός τρισδιάστατου χώρου στην οθόνη. Μία άλλη αρκετά πιο ακριβή τεχνική, η

[1] Βεβαίως, κάποιοι από αυτούς τους υπολογισμούς μπορεί να εκτελούνται και στις σύγχρονες κάρτες γραφικών ή σε άλλο ειδικό hardware. Παράδειγμα του πρώτου αποτελεί η μηχανής φυσικής PhysX ενώ παράδειγμα του δεύτερου αποτελούν οι PPUs (Physics Processing Units).

[2] Δεδομένου ότι οι οθόνες είναι διακριτές το στάδιο της ραστεροποίησης πάντα θα υπάρχει. Εδώ όμως διαχωρίζουμε την προσέγγιση αυτή ως τάση να μεταφέρονται όλοι οι υπολογισμοί σε αυτό το στάδιο (αντί στο επίπεδο της γεωμετρίας και της εφαρμογής) για λόγους απόδοσης.

οποία όμως παρέχει πιο ρεαλιστικά αποτελέσματα είναι η ανίχνευση ακτίνας (Ray Tracing). Μέσω της ραστεροποίησης κάθε σημείο ενός αντικειμένου μεταφράζεται σε ένα εικονοστοιχείο στην οθόνη. Αυτή η τεχνική είναι αρκετά αποδοτική και φέρνει ικανοποιητικά αποτελέσματα, αλλά είναι μόνο μία προβολή του χώρου χωρίς να λαμβάνει υπόψη της φυσικές ιδιότητες του φωτός ως προς την ανάκλαση. Η ανίχνευση ακτίνας επιχειρεί να προσεγγίσει τη λειτουργία του φωτός στον πραγματικό κόσμο, “πετάγοντας” ακτίνες από το μάτι της κάμερας στον χώρο θέασης και ανακαλώντας τες στις επιφάνειες που συναντάει.



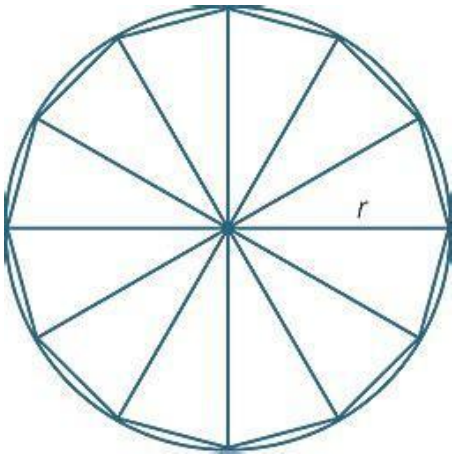


Αν και τα αποτελέσματα είναι εντυπωσιακά, ακόμα και οι σημερινές κάρτες γραφικών δυσκολεύονται να εφαρμόσουν αυτή τη τεχνική αποδοτικά (να θυμίσουμε ότι θέλουμε όλη η διαδικασία να ολοκληρώνεται τουλάχιστον 60 φορές το δευτερόλεπτο). Ωστόσο με τη χρήση των τεχνικών DLSS και FSR των NVidia και AMD αντίστοιχα, παιχνίδια που την υποστηρίζουν μπορούν να ενεργοποιήσουν ray tracing και να τρέχουν ικανοποιητικά με το μόνο κόστος να αφορά μία μικρή μείωση της ευκρίνειας της εικόνας. Τόσο το DLSS όσο και το FSR πετυχαίνουν αυτή την ώθηση στην αποδοτικότητα αποδίδοντας το παιχνίδι σε μία χαμηλότερη ανάλυση (για παράδειγμα 720p) και μετά χρησιμοποιούν αλγορίθμους μεγέθυνσης για να φανεί η εικόνα στην επιθυμητή ανάλυση (για παράδειγμα 1080p). Φαίνεται ότι το υλικό δεν είναι ακόμα έτοιμο να εκμεταλλευτεί την τεχνική ανίχνεσυςη ακτίνας για απόδοση γραφικών σε πραγματικό χρόνο, ωστόσο τόσο το DLSS όσο και το FSR φέρνουν τόσο εντυπωσιακά αποτελέσματα, που ίσως η αναβάθμιση του υλικού να μην είναι απαραίτητη. Ακόμα και σύγχρονες κονσόλες όπως το PlayStation5 Pro χρησιμοποιούν τέτοιου είδους τεχνολογία για να αποδώσουν σύγχρονα παιχνίδια σε υψηλότερες αναλύσεις.

Πριν το στάδιο της εφαρμογής..

Προκειμένου να μπορεί η μηχανή ανάπτυξης να επεξεργαστεί τα στοιχεία του παιχνιδιού, όπως τα μοντέλα, τα sprites, τη μουσική, και άλλα, πρέπει να είναι σε κάποια αναγνωρίσιμη μορφή. Συνήθως δεν είναι δόκιμο να φτιάξουμε κάποια δικιά μας μορφή καθώς αυτό απαιτεί την ανάπτυξη ενός προγράμματος ψηφιακής δημιουργίας περιεχομένου το οποίο παράγει στοιχεία στην μορφή που επιθυμούμε. Αυτό είναι χρονοβόρο όχι μόνο λόγω του χρόνου που απαιτείται για την υλοποίηση τέτοιων προγραμμάτων, αλλά και λόγω του χρόνου που απαιτείται για την εκπαίδευση του προσωπικού στα καινούργια προγράμματα. Είναι καλύτερο να χρησιμοποιούν οι καλλιτέχνες εφαρμογές τις οποίες γνωρίζουν καλά και να μετατρέπουμε τη μορφή του κάθε στοιχείου σε μία αναγνωρίσιμη μορφή από την μηχανή παιχνιδιού. Το σύνολο διαδικασιών που επιβάλουν τη συμβατότητα του περιεχομένου με τη μηχανή παιχνιδιών ονομάζεται ροή προετοιμασίας περιεχομένου (Asset Conditioning Pipeline). Εκτός από τη μετατροπή της μορφής του περιεχομένου, μέρος της ACP είναι και η κατασκευή δομών δεδομένων οι οποίες χρησιμοποιούνται στην μηχανή, όπως μία δομή για το γράφημα σκηνής. Επιπλέον, μπορεί να έχουμε μία πολύπλοκη σκηνή με διάφορες πηγές φωτός, αλλά η σκηνή να μην αλλάζει μορφή (τα αντικείμενα μένουν στατικά, τα φώτα δεν μπορούν να σβήσουν). Σε αυτή την περίπτωση μπορούμε να προϋπολογίσουμε τον φωτισμό και τις σκιές και να τις αποθηκεύσουμε. Έτσι όταν φορτώσουμε αυτή τη σκηνή στο παιχνίδι, αρκεί να φορτώσουμε τον ήδη έτοιμο φωτισμό, χωρίς να χρειάζεται να κάνουμε ακριβούς υπολογισμούς φωτισμού. Η διαδικασία αυτή ονομάζεται baking. Αν και αυτή η τεχνική χρησιμοποιείται συνήθως για τον φωτισμό μίας σκηνής, μπορούμε να “ψήσουμε” οτιδήποτε το οποίο παραμένει το ίδιο ανεξαρτήτως συνθηκών. Ας υποθέσουμε ότι έχουμε ένα κτήριο το οποίο θέλουμε να καταρρεύσει και να φαίνεται ρεαλιστικό θα μπορούσαμε να τρέξουμε μία προσομοίωση κατάρρευσης και να αποθηκεύσουμε το αποτέλεσμα ως ένα animation.

Αντικείμενα και Πολυγωνικά Πλέγματα



Έχουμε αναφέρει αρκετές φορές μέχρι τώρα τον όρο “μοντέλο” όταν περιγράφουμε ένα αντικείμενο στον τρισδιάστατο χώρο. Αλλά τι είναι πραγματικά ένα μοντέλο; Ένα μοντέλο είναι ένα πολυγωνικό πλέγμα, δηλαδή ένα σύνολο από πολύγωνα, συνήθως τρίγωνα και κάποιες φορές τετράπλευρα, τα οποία θυμίζουν τη μορφή του αντικειμένου ή χαρακτήρα που μοντελοποιούμε. Οι σύγχρονες κάρτες γραφικών μάλιστα, έχουν σχεδιαστεί με τέτοιο τρόπο ώστε να μπορούν να αποδίδουν τρίγωνα όσο το δυνατόν πιο αποτελεσματικά.

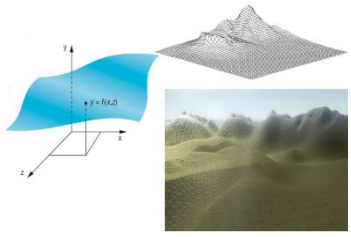
Γιατί επιλέξαμε το τρίγωνο ως το βασικό σχήμα ενός τρισδιάστατου μοντέλου; Προφανώς επειδή είναι το φθηνότερο και απλούστερο σχήμα το οποίο μπορεί να αποδώσει μορφή στις τρεις διαστάσεις. Για να ορίσουμε ένα επίπεδο στον τρισδιάστατο χώρο το τρίγωνο αρκεί. Αντιθέτως, αν χρησιμοποιούσαμε ένα τετράπλευρο, τότε δεν είναι απαραίτητο ότι και οι τέσσερις κορυφές θα βρισκόντουσαν στο ίδιο επίπεδο. Επιπροσθέτως χρησιμοποιώντας τρίγωνα μπορούμε εύκολα να τριγωνοποιήσουμε οποιοδήποτε πολύγωνο. Ίσως όχι με τέλεια λεπτομέρεια, αλλά δεν αναζητούμε την τελειότητα στα βιντεοπαιχνίδια, εξάλλου το πόσο ευχάριστο είναι ένα παιχνίδι έχει μεγαλύτερη σημασία από το πόσο κοντά στην πραγματικότητα είναι. Ένα ακόμα πλεονέκτημα των τριγώνων είναι ότι μπορούν να αναπαραστήσουν εξαιρετικά λεπτά χαρακτηριστικά επιφανειών, όπως τη λεπίδα ενός μαχαιριού ή ένα καρφί. Ωστόσο δεν προτείνεται η χρήση πολύ λεπτών πολυγώνων. Ας μην ξεχάσουμε ότι δουλεύουμε με υπολογιστές και ότι η θέση της κάθε κορυφής αποθηκεύεται σε ένα float ή double, συνεπώς είναι ευάλωτα σε σφάλματα στρογγυλοποίησης.

Παρά τις ευκολίες που μας παρέχονται, η χρήση πολυγώνων για την μοντελοποίηση αντικειμένων δεν είναι το τέλειο σύστημα. Κύριο μειονέκτημα αυτής της προσέγγισης είναι το υψηλό κόστος τόσο σε μνήμη όσο και επεξεργασία. Για την αποθήκευση ενός πλέγματος δεν αρκεί η αποθήκευση της κάθε κορυφής, απαιτείται και η αποθήκευση των ακμών του κάθε τριγώνου. Ας πάρουμε για παράδειγμα το παραπάνω σχήμα. Η κορυφή στο κέντρο του κύκλου την οποία χρησιμοποιούν όλα τα τρίγωνα δεν χρειάζεται να αποθηκευτεί μία φορά για κάθε τρίγωνο. Άλλο μειονέκτημα είναι η μοντελοποίηση καμπυλωτών επιφανειών, όπως ο κύκλος της παραπάνω εικόνας. Για τέτοιου είδους επιφάνειες απαιτείται μία προσεγγιστική μοντελοποίηση. Συνεπώς, αφού δεν έχουμε την υπολογιστική ισχύ να αναπαραστήσουμε άπειρα πολύγωνα για κάθε αντικείμενο, θα πρέπει να θυσιάσουμε λίγο ρεαλισμό για να

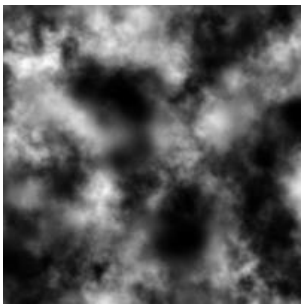
βεβαιωθούμε ότι το παιχνίδι θα μπορεί να τρέχει στα σύγχρονα αλλά όχι υψηλών επιδόσεων συστήματα.

Επαναλαμβάνουμε ότι ένα πλέγμα είναι ένα δίκτυο τριγώνων, τα οποία συνδέονται μεταξύ τους μέσω κοινών κορυφών και ακμών με στόχο να σχηματίσουν μία ενιαία συνεχή επιφάνεια η οποία έχει τη μορφή του αντικειμένου που μοντελοποιούμε. Ένα πλέγμα είναι πολύ πιο διαχειρίσιμο και αποτελεσματικό από μία συλλογή τριγώνων τα οποία αποθηκεύονται ανεξάρτητα το ένα με το άλλο. Οι ελάχιστες πληροφορίες που απαιτούνται για την αποθήκευση ενός τριγωνικού πλέγματος είναι το σύνολο των τριγώνων, δηλαδή των κορυφών, και οι θέσεις τους στον τρισδιάστατο χώρο. Ωστόσο πολλές μορφές αρχείων αποθηκεύουν πρόσθετα δεδομένα όσον αφορά τις κορυφές, ακμές, και όψεις, τα οποία χρησιμοποιούνται για τη χαρτογράφηση υφής, σκίασης, animations, και άλλες λειτουργίες οι οποίες δίνουν στα μοντέλα μας χαρακτήρα.

Πεδία Ύψους



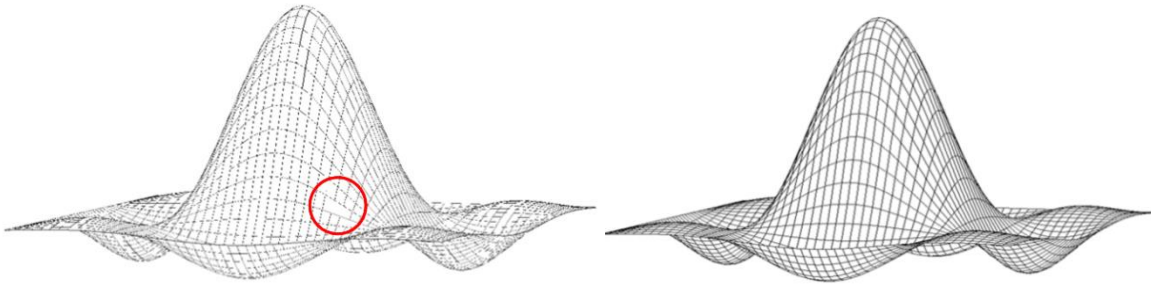
Για μεγάλες επιφάνειες, όπως ένας ανοιχτός κόσμος, είναι εξαιρετικά ακριβό να αποθηκεύουμε τη θέση του κάθε τριγώνου. Αντί αυτού, πολλά παιχνίδια χρησιμοποιούν έναν χάρτη ύψους ο οποίος ορίζει τιμή ύψους για κάθε μοναδική τιμή x και z. Ένα παράδειγμα τέτοιου χάρτη φαίνεται παρακάτω. Μπορούμε να φανταστούμε την παρακάτω εικόνα σαν έναν γεωμορφολογικό χάρτη όπου το μαύρο δηλώνει το ελάχιστο ύψος που επιτρέπεται να έχει η επιφάνεια ενώ το λευκό δηλώνει το μέγιστο ύψος. Έτσι όταν θέλουμε να φτιάξουμε μία μεγάλη πεδιάδα με διαφορά ύψους, αρκεί να φτιάξουμε ένα επίπεδο και να εφαρμόσουμε τον χάρτη ύψους. Το κύριο μειονέκτημα αυτής της τεχνικής είναι ότι δεν επιτρέπεται η ίδια θέση (x, z) να έχει πάνω από μία τιμή y. Ωστόσο τίποτα δεν μας εμποδίζει από το να χρησιμοποιήσουμε αυτή την τεχνική για τη γενική πεδιάδα και να την εμπλουτίσουμε με πιο πολύπλοκα μοντέλα τα οποία αποθηκεύονται κανονικά.



Κρυμμένες Γραμμές και Όψεις

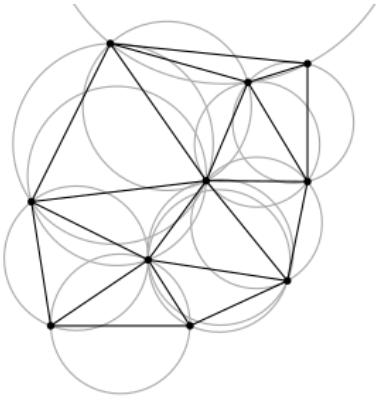
Για να σχεδιάσουμε ένα τρισδιάστατο πλέγμα σχεδιάζουμε τις κορυφές του. Ωστόσο καταλαβαίνετε ότι δεν θα είναι όλες οι κορυφές πάντα ορατές. Κάποιες μπορεί να μη φαίνονται καθώς βρίσκονται πίσω από άλλες κορυφές οι οποίες βρίσκονται πιο κοντά στην κάμερα. Συνεπώς δεν υπάρχει λόγος να τις ζωγραφίσουμε. Έτσι μειώνουμε την

υπολογιστική ισχύ που απαιτείται αγνοώντας κορυφές οι οποίες σχηματίζουν τρίγωνα τα οποία κρύβονται πίσω από άλλα τρίγωνα. Όταν επιλέξουμε τα ορατά τρίγωνα ζωγραφίζουμε τις κορυφές τους και στη συνέχεια τις ακμές τους. Αν και ζωγραφίσαμε τα τρίγωνα και τις γραμμές σωστά, υπάρχει ακόμα η πιθανότητα μικρών αριθμητικών λαθών τα οποία επιβάλλουν σε κάποια τμήματα των γραμμών να βρίσκονται πίσω από τα αντίστοιχα τμήματα του τριγώνου. Ένα τέτοιο σφάλμα φαίνεται στην πρώτη εικόνα από κάτω. Χρησιμοποιώντας την τεχνική μετατόπισης πολυγώνου το σφάλμα εξαφανίζεται.



Θέματα με Πλέγματα

Για την κατασκευή πλεγμάτων από ένα σύνολο σημείων χρησιμοποιούμε προγράμματα κατασκευής μοντέλων στα οποία μπορούμε να ζωγραφίσουμε το κάθε τρίγωνο και ακμή με το χέρι, να ξεκινήσουμε από ένα απλό σχήμα και να το επεκτείνουμε, ή να αρχίσουμε να ζωγραφίζουμε με μία βούρτσα χώρου και να αφήσουμε το πρόγραμμα να σχεδιάσει τα τρίγωνα αυτόματα. Αν και η τελευταία τεχνική συνήθως αποφεύγεται στον τομέα των βιντεοπαιχνιδιών λόγο των υπερβολικά πυκνών πλεγμάτων που δημιουργεί, αξίζει να δούμε πως λειτουργεί σε χαμηλό επίπεδο. Ανάμεσα σε άλλους αλγορίθμους, μία γνωστή τεχνική είναι η τριγωνοποίηση Delaunay. Σύμφωνα με αυτή την τεχνική, ένα σύνολο σημείων σε ένα επίπεδο υποδιαιρείται σε τρίγωνα έτσι ώστε κάθε κύκλος, ο οποίος τέμνει και τις τρεις κορυφές ενός τριγώνου, δεν περιέχει κανένα από τα σημεία αυτά. Η τεχνική αυτή μεγιστοποιεί το μέγεθος της μικρότερης γωνίας του κάθε τριγώνου, αποφεύγοντας έτσι τρίγωνα με υπερβολικά οξείες γωνίες (όσο το δυνατόν περισσότερο).

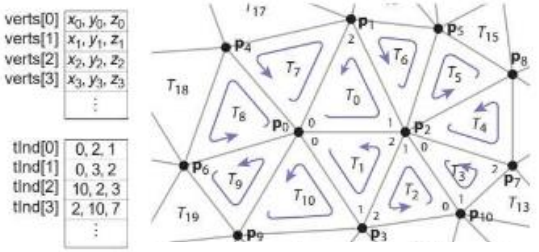


Μία άλλη εφαρμογή η οποία έχει δει πρόοδο σήμερα είναι η σάρωση αντικειμένων του πραγματικού κόσμου χρησιμοποιώντας κάποια κάμερα και η αυτόματη ψηφιοποίησή τους. Εφαρμογές όπως το Reality Scan της Unreal επιχειρούν να απλοποιήσουν την δημιουργία τρισδιάστατων μοντέλων επιτρέποντας στον χρήστη να τραβήξει φωτογραφίες ενός

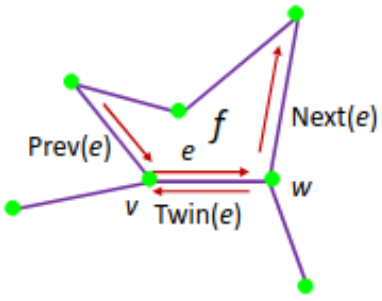
αντικειμένου από διάφορες οπτικές γωνίες, τις οποίες η εφαρμογή χρησιμοποιεί για να δημιουργήσει ένα πλέγμα το οποίο αναπαριστά το αντικείμενο. Αν και τα αποτελέσματα ποικίλλουν, η τεχνολογία είναι αρκετά εντυπωσιακή.

Δομή Δεδομένων για Πλέγμα

Μία από τις πιο γνωστές δομές δεδομένων για πλέγματα είναι το δεικτοδοτούμενο σύνολο όψεων (.obj). Η μορφή αυτή είναι ουσιαστικά μία λίστα από κορυφές όπου για κάθε μία αποθηκεύεται η θέση της στον τρισδιάστατο χώρο. Ακολουθεί μία λίστα από πολύγωνα όπου η κάθε τιμή αναφέρεται στην αντίστοιχη κορυφή. Επιτρέπεται ένα πολύγωνο να αντιστοιχεί σε πάνω από μία κορυφές, συνεπώς δεν είμαστε περιορισμένοι μόνο σε τρίγωνα. Όπως βλέπουμε την εξωτερική όψη των πολυγώνων, οι αντίστοιχες κορυφές παρατίθενται με σειρά αντίστροφη του ρολογιού.



Μία άλλη δομή δεδομένων είναι ο διπλοσυνδεδεμένος κατάλογος ακμών. Με την δομή αυτή αποθηκεύεται τόσο γεωμετρική όσο και τοπολογική πληροφορία. Κάθε ακμή έχει δύο ημιακμές με αντίθετη φορά, έναν δείκτη προς την επόμενη ακμή και έναν δείκτη προς την προηγούμενη ακμή. Στο διπλανό σχήμα η ακμή e έχει αφετηρία την v και απόληξη τη w, ενώ για την αντίθετη ακμή $Twin(e)$ ισχύει ότι η w είναι η αφετηρία και v η απόληξη. Για κάθε έδρα f αποθηκεύεται ένας δείκτης σε μία αυθαίρετη ημιακμή στο σύνορό της. Από αυτή την ημιακμή το σύστημα διαπερνά το σύνορο της έδρας με φορά αντίθετη του ρολογιού. Αυτή η αναπαράσταση αποθηκεύει περισσότερη πληροφορία, αλλά είναι προφανώς πιο ακριβή.

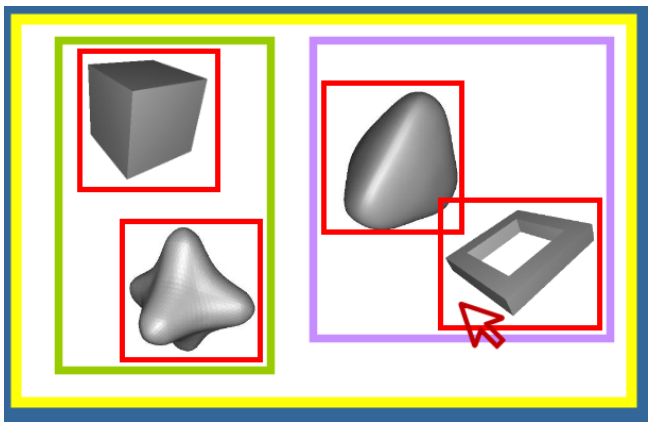


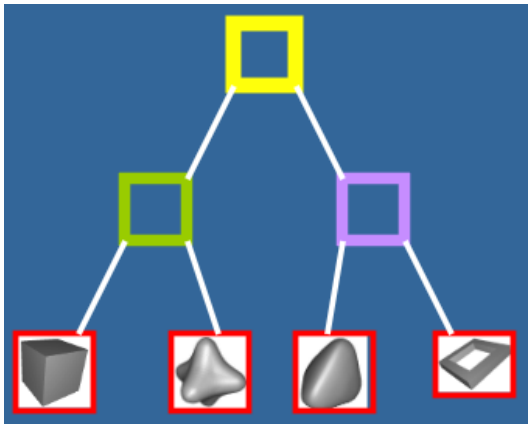
Γράφημα Σκηνής

Γνωρίζουμε ότι ένας χώρος στο παιχνίδι μας αποτελείται από εκατοντάδες, αν όχι χιλιάδες διαφορετικά αντικείμενα. Ο παίκτης μπορεί ανά πάσα στιγμή να βλέπει μόνο ένα μικρό πλήθος αυτών, αλλά αυτό δεν σημαίνει ότι τα αντικείμενα τα οποία βρίσκονται έξω από τον χώρο θέασης δεν υπάρχουν. Γνωρίζουμε ότι για να μειώσουμε την απαιτούμενη υπολογιστική ισχύ δεν αποδίδουμε τα αντικείμενα τα οποία βρίσκονται έξω από τον χώρο

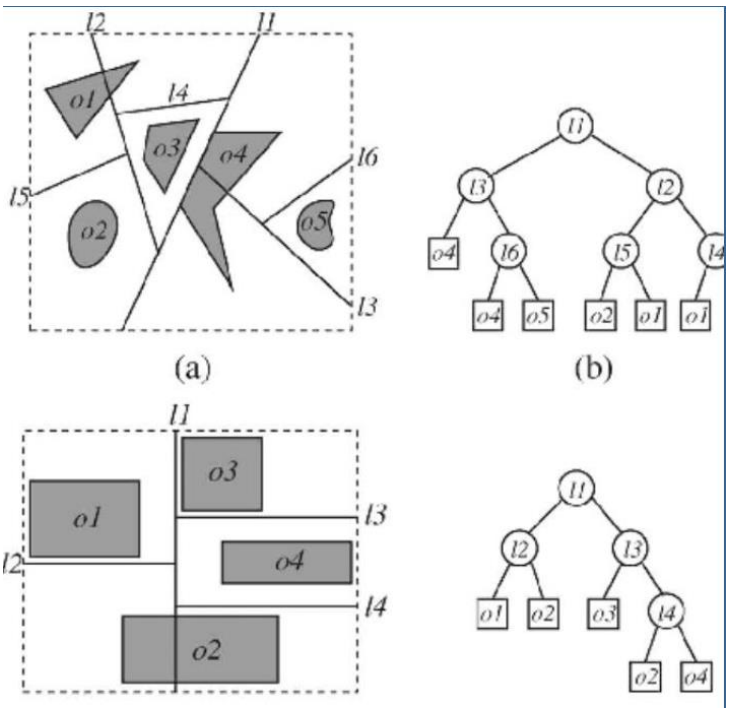
θέασης, όμως το να ελέγξουμε κάθε αντικείμενο ένα-ένα δεν είναι πολύ φθηνό. Για αυτό χρησιμοποιούμε ένα γράφημα σκηνής, ή μία χωρική δομή δεδομένων.

Τέτοιες δομές οργανώνουν τη γεωμετρία (δύο, τριών, ή παραπάνω διαστάσεων, αλλά εμείς ενδιαφερόμαστε για 2d και 3d). Όπως είπαμε ο στόχος τους είναι η ταχύτερη επεξεργασία του χώρου. Επιταχύνουν τον χρόνο απόδοσης, τον χρόνο ελέγχου τομής, την ανίχνευση συγκρούσεων, καθώς και τις μεθόδους απόδοσης ray tracing και καθολικού φωτισμού. Πως τα επιτυγχάνουν όλα αυτά; Οργανώνοντας την γεωμετρία με ιεραρχικό τρόπο. Ας θεωρήσουμε το παρακάτω σχήμα. Όταν ο χρήστης κάνει κλικ σε κάποιο σημείο της οθόνης θέλουμε να επιλέξουμε το αντίστοιχο αντικείμενο. Η δομή δεδομένων φαίνεται με το δέντρο της δεξιάς εικόνας. Όταν κάνουμε κλικ στην οθόνη αντί να ελέγξουμε για κάθε αντικείμενο ξεχωριστά, ελέγχουμε για τα παιδιά της ρίζας. Έτσι ξέρουμε ότι κάναμε κλικ εντός της μωβ περιοχής, συνεπώς δεν χρειάζεται να ελέγξουμε για τα αντικείμενα της πράσινης περιοχής. Στη συνέχεια ελέγχουμε τα παιδιά της μωβ περιοχής μέχρι να καταλήξουμε στο ότι κάναμε κλικ στο παράθυρο. Στην πράξη ο χρόνος ανίχνευσης μειώνεται από $O(n)$ σε $O(\log n)$.





Μία από αυτές τις δομές είναι η ιεραρχία οριοθετημένου όγκου (Bounding Volume Hierarchy). Σύμφωνα με τη δομή αυτή κάθε αντικείμενο περικλείεται από απλούς γεωμετρικούς όγκους (οι οποίοι ονομάζονται BV για Bounding Volume), όπως σφαίρες και ορθογώνια παραλληλεπίπεδα, τα οποία με τη σειρά τους περικλείονται από μεγαλύτερους όγκους. Έτσι σχηματίζεται ένα k-αδικό δέντρο στο οποίο τα φύλλα έχουν την πραγματική γεωμετρία την οποία μπορεί να δει ο παίκτης, ενώ οι υπόλοιποι κόμβοι χρησιμοποιούνται μόνο για την οργάνωση της βάσης. Θυμίζουμε ότι ο όγκος του κάθε κόμβου πρέπει να εσωκλείει τη γεωμετρία όλων των παιδιών του, οπότε, στο παραπάνω σχήμα, αν ο κύβος



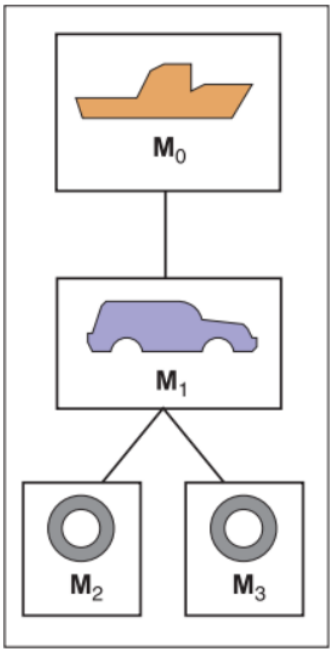
ήταν παιδί του μωβ κόμβου, τότε ο γεωμετρικός όγκος του μωβ κόμβου θα έπρεπε να είναι αρκετά μεγάλος για να τον εμπεριέχει.

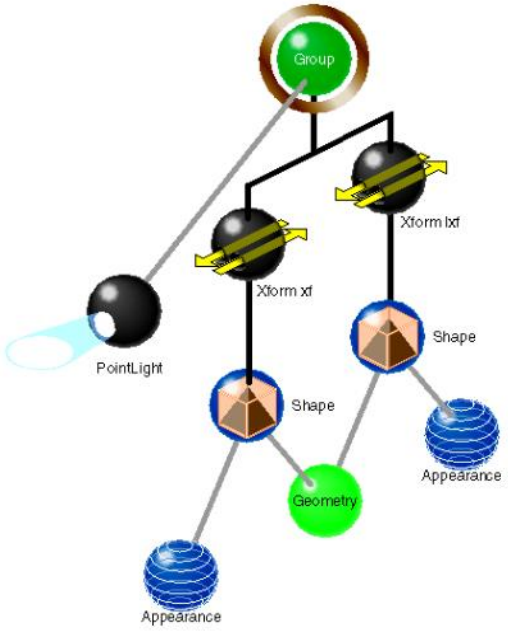
Μία άλλη δομή είναι το δυαδικό δέντρο χωρισμού χώρου ή Binary Space Partitioning Tree. Η βασική λογική πίσω από αυτή τη δομή είναι η ίδια, ωστόσο ο τρόπος με τον οποίο χωρίζει τα αντικείμενα και σχηματίζει το δέντρο είναι διαφορετικός. Σύμφωνα με τα BSP δέντρα, ο χώρος διαμερίζεται με ένα επίπεδο, και μετά η γεωμετρία τοποθετείται στον υποχώρο στον οποίο ανήκει. Το ίδιο γίνεται αναδρομικά για όλους τους υποχώρους. Ο διαχωρισμός μπορεί να γίνει ευθυγραμμισμένος ως προς τους άξονες του χώρους, ή ως προς τα αντικείμενα. Είναι αξιοσημείωτο ότι χρησιμοποιώντας αυτή την δομή είναι πιθανό κάποια αντικείμενα να χωριστούν σε υπομέρη καθώς η διαμέριση περνάει από μέσα τους. Αυτό σημαίνει ότι ένα αντικείμενο μπορεί να ανήκει σε δύο ή περισσότερους υποχώρους, ή πιο σωστά, μέρος του αντικειμένου θα ανήκει στον έναν υπόχωρο, ενώ άλλο μέρος του θα ανήκει σε έναν άλλο υποχώρο. Αυτό δεν είναι πρόβλημα στην πράξη καθώς στην πραγματικότητα δεν αναφερόμαστε σε χώρους, αλλά σε κορυφές, οι οποίες είναι σημεία.

Η δομή BVH είναι η πλέον χρησιμοποιούμενη καθώς είναι εύκολη να κατανοηθεί και να υλοποιηθεί. Το μειονέκτημα όμως είναι ότι αποθηκεύει μόνο γεωμετρία, δηλαδή κορυφές, η οποία δεν είναι αρκετή για την απόδοση γραφικών. Ένα πραγματικό γράφημα σκηνής είναι ένα εμπλουτισμένο BVH, το οποίο εκτός από γεωμετρία αποθηκεύει επίσης φώτα, υφές, μετασχηματισμούς, και άλλα στοιχεία.

Η εύρεση γεωμετρίας δεν είναι ο μόνος ρόλος του γραφήματος σκηνής. Βοηθάει ιδιαίτερα και σε άλλα κομμάτια της απόδοσης γραφικών. Για παράδειγμα, ας υποθέσουμε ότι υπάρχει ένα καράβι, το οποίο έχει μέσα του ένα αυτοκίνητο. Το καράβι και το αυτοκίνητο μπορούν να είναι διαφορετικά αντικείμενα στον χώρο. Τότε όταν κινείται το καράβι πρέπει να κινηθεί αντίστοιχα και το αυτοκίνητο. Οπότε αν το καράβι μετασχηματίζεται από το μητρώο M_0, τότε το αυτοκίνητο θα μετασχηματίζεται από το μητρώο $M_0 M_1$. Επιπροσθέτως το αυτοκίνητο μπορεί να μην είναι ένα ενιαίο αντικείμενο, αλλά να αποτελείται από το σώμα του

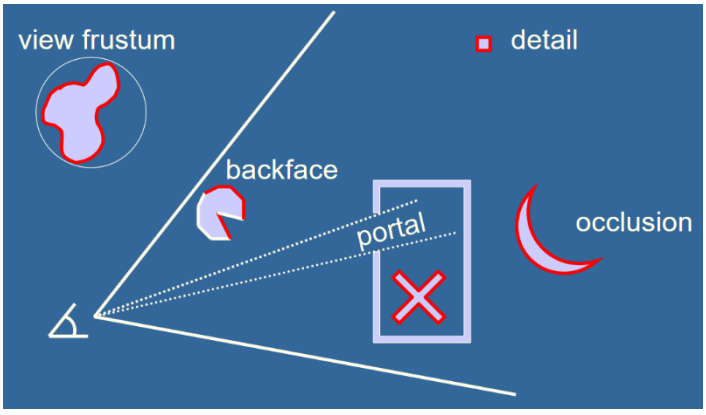
αυτοκινήτου, την αριστερή του ρόδα, και την δεξιά του ρόδα (από τη μία πλευρά). Τότε το σώμα θα μετασχηματίζεται από το μητρώο M_0M_1, ενώ οι ρόδες θα μετασχηματίζονται από τα μητρώα $M_0M_1M_2$ και $M_0M_1M_3$. Αντίστοιχα για φωτισμό, χρώμα, τύπος αντικειμένου, υλικό, και άλλα στοιχεία, το κάθε αντικείμενο θα μετασχηματίζεται εξαρτημένο από τα αντικείμενα γονείς του.



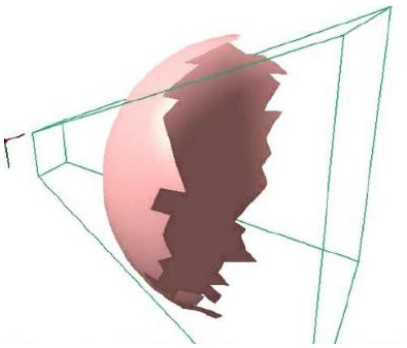


Αποκοπή Γεωμετρικών Αντικειμένων

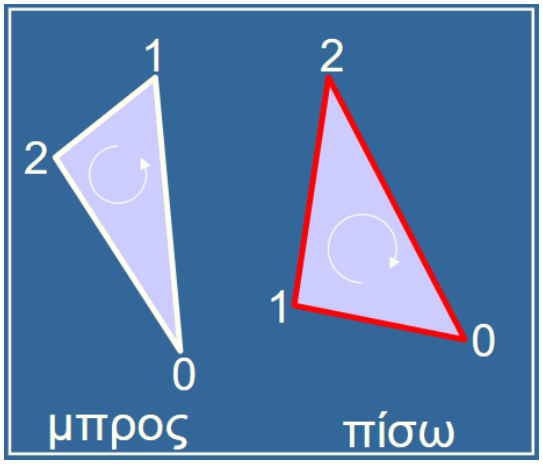
Έχουμε αναφέρει ήδη ότι δεν αποδίδουμε γεωμετρία η οποία βρίσκεται εκτός του χώρου θέασης, αλλά αυτή δεν είναι η μόνη μορφή αποκοπής. Ας μιλήσουμε με λίγη παραπάνω λεπτομέρεια για τεχνικές αποκοπής. Να θυμίσουμε επίσης ότι δεν μιλάμε για ολόκληρα αντικείμενα, αλλά για κορυφές και πολύγωνα.

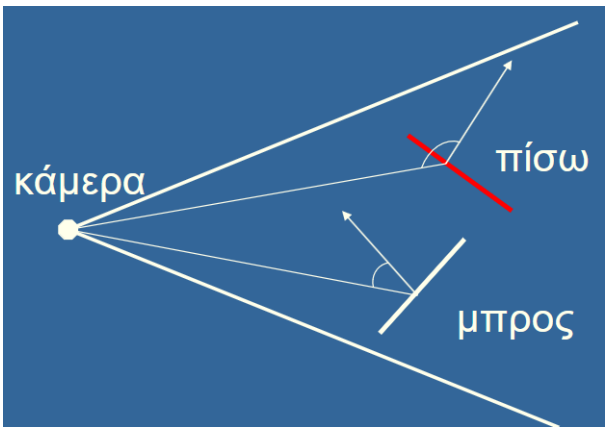


- **View Frustum**: Ο πρώτος τύπος τεχνικής αποκοπής είναι η αποκοπή πεδίου θέασης, την οποία έχουμε δει ήδη, απλώς αγνοούμε τις κορυφές οι οποίες βρίσκονται έξω από τον χώρο θέασης. Εδώ βοηθούν πολύ οι χωρικές δομές δεδομένων που εξετάσαμε προηγουμένως.



- **Detail**: Ένας άλλος τύπος είναι η αποκοπή λεπτομέρειας. Σύμφωνα με αυτόν τον τύπο, τα τρίγωνα τα οποία βρίσκονται εντός του χώρου θέασης αλλά είναι τόσο μικρά που δεν φαίνονται μπορούν να αγνοηθούν.
- **Backface**: Γνωρίζουμε ότι ένα αντικείμενο στον χώρο δεν είναι μόνο ένα πλέγμα από σημεία, αλλά έχει και υφές που δίνουν εικόνα και χρώμα στο πλέγμα μας. Ως αποτέλεσμα κάποια πολύγωνα μπορεί να καλύπτονται από κάποια άλλα. Για παράδειγμα, αν έχουμε το μοντέλο ενός ανθρώπου και το βλέπουμε από μπροστά, τότε δεν θα μπορούμε να δούμε το πίσω μέρος του κεφαλιού του, καθώς καλύπτεται από το πρόσωπό του. Συνεπώς δεν υπάρχει λόγος να αποδώσουμε τα πιο “πίσω” πολύγωνα τα οποία καλύπτονται από τα πιο “μπροστά”. Αυτή η τεχνική υλοποιείται ελέγχοντας τον προσανατολισμό των πολυγώνων και συγκρίνοντάς τον με τον προσανατολισμό της κάμερας. Αν ένα πολύγωνο είναι προσανατολισμένο σε αντίθετη κατεύθυνση από αυτή του θεατή τότε αγνοείται. Θυμίζουμε ότι αποθηκεύουμε όλες τις κορυφές και πολύγωνα έτσι ώστε οι αριθμημένες κορυφές ενός πολυγώνου να έχουν φορά αντίθετη του ρολογιού. Έτσι εξετάζοντας τη φορά ενός πολυγώνου μπορούμε να υπολογίσουμε γρήγορα τον προσανατολισμό του.



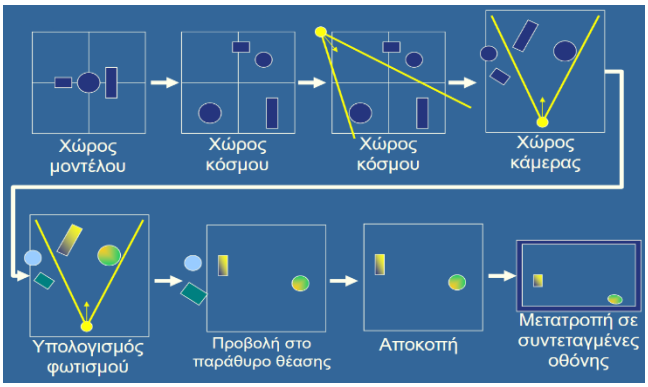


- **Occlusion**: Οι τύποι portal και occlusion έχουν πρακτικά τον ίδιο ρόλο. Έχουν παρόμοια λειτουργία με το backface, με την διαφορά ότι δεν εξετάζουμε μόνο ένα μοντέλο ή αντικείμενο, αλλά την κάλυψη ενός μοντέλου από τα υπόλοιπα μοντέλα. Μπορεί για παράδειγμα να έχουμε το μοντέλο ενός ανθρώπου το οποίο να καλύπτεται κατά το ήμισυ από έναν τοίχο. Σε αυτή την περίπτωση θα πρέπει να αγνοήσουμε τα πολύγωνα που καλύπτονται από τον τοίχο. Η αποκοπή απόκρυψης είναι ένα πολύ δύσκολο πρόβλημα στο οποίο έχουν γίνει πολλές έρευνες και δυστυχώς δεν έχουμε τον χρόνο να το αναλύσουμε περαιτέρω.
- **Portal**: Είναι μία ειδική περίπτωση της γενικής απόκρυψης που χρησιμοποιείται για να αυξήσει την απόδοση αποκρύπτοντας γρήγορα αντικείμενα που δεν φαίνονται μέσα από μία πύλη σε ένα άλλο χώρο. Ένα τέτοιο παράδειγμα πύλης θα μπορούσε να είναι η πόρτα προς ένα άλλο δωμάτιο ή ένας μικρός χώρος μεταξύ ορεινών όγκων.

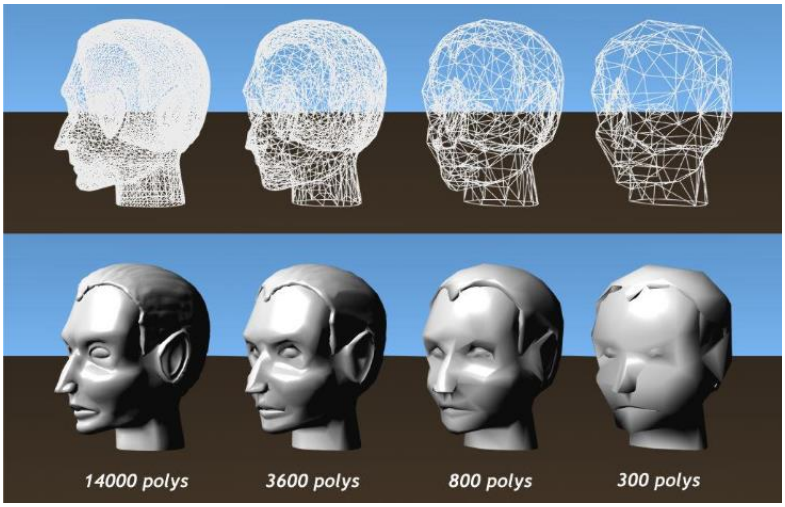
Απόδοση Γραφικών και Τεχνικές Μείωσης Υπολογιστικών Πόρων

Έχουμε μιλήσει ήδη αρκετά για την απόδοση γεωμετρίας. Χρησιμοποιούμε μετασχηματισμούς μέσω μητρώων για μετακίνηση, περιστροφή και κλιμάκωση από το χώρο του μοντέλου στο χώρο του κόσμου. Το ίδιο γίνεται και για την κάμερα. Τα παιδιά στο δέντρο του γραφήματος σκηνής θα κληρονομήσουν και τους μετασχηματισμούς που εφαρμόζονται στους γονείς τους. Όταν γίνουν όλοι οι μετασχηματισμοί περνάμε στο πεδίο της αποκοπής, για το οποίο μόλις μιλήσαμε. Και τέλος προβάλουμε το χώρο στην οθόνη. Υπάρχει, όμως, ένα κρίσιμο στοιχείο που δεν έχουμε αναφέρει ακόμα, τον φωτισμό. Ένας χώρος τριών διαστάσεων απαιτεί κάποιο σύστημα φωτισμού για να αποδοθεί με τέτοιο τρόπο ώστε να φαίνεται όμορφος στο χρήστη. Το πρόβλημα είναι ότι ρεαλιστικές αποδόσεις φωτισμού είναι υπολογιστικά ακριβές. Αν φτιάχναμε μία ταινία αυτό δεν θα ήταν πρόβλημα, καθώς πρέπει να αποδοθεί η κάθε εικόνα μία φορά και μετά να αποθηκευτεί σε μορφή βίντεο, ωστόσο εμείς θέλουμε να αποδώσουμε γραφικά σε πραγματικό χρόνο, συχνά τουλάχιστον 60 φορές το δευτερόλεπτο. Επιπλέον, θέλουμε το παιχνίδι μας να λειτουργεί σε όσο το δυνατόν περισσότερα μηχανήματα, όχι μόνο στα πιο ισχυρά. Για αυτό

χρησιμοποιούμε τεχνικές οι οποίες μιμούνται τη συμπεριφορά του φωτός στη φύση. Αυτές οι τεχνικές είναι από τη πλευρά της Φυσικής λάθος αλλά δίνουν εξαιρετικά αποτελέσματα.

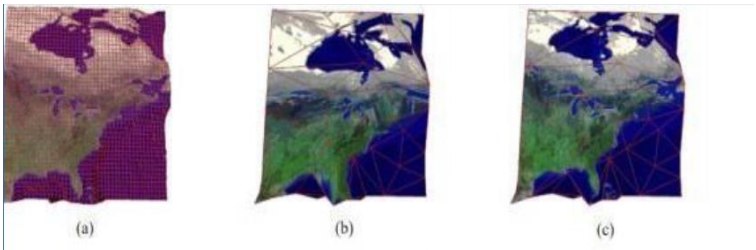


Μία άλλη τεχνική μείωσης υπολογιστικών πόρων είναι η απόδοση μοντέλων σε διαφορετικά επίπεδα λεπτομέρειας. Όταν ο χαρακτήρας τον οποίο ελέγχουμε βρίσκεται κοντά στην κάμερα είναι σημαντικό να έχει όσο περισσότερη λεπτομέρεια. Ωστόσο όσο απομακρύνεται από την κάμερα τόσο λιγότερη από αυτή την λεπτομέρεια θα είναι ορατή στον παίκτη. Συνεπώς δεν υπάρχει λόγος να επενδύουμε υπολογιστική ισχύ σε κάτι το οποίο δεν θα είναι ορατό, για αυτό όταν κάποιο μοντέλο απομακρύνεται αρκετά από την κάμερα το αλλάζουμε σε ένα παρόμοιο με λιγότερα πολύγωνα. Το LoD (Level of Detail) δεν εφαρμόζεται μόνο στα μοντέλα, αλλά είναι ένας γενικότερος όρος που αναφέρεται στην ιδέα της αφαίρεσης μη ορατής από τον παίκτη λεπτομέρειας. Άλλα παραδείγματα αποτελούν την αλλαγή μίας υφής σε μία χαμηλότερης ανάλυσης ή την μείωση των FPS ενός animation όταν το αντίστοιχο μοντέλο βρίσκεται μακριά. Στην LoD απόδοση ανήκει και ο τύπος αποκοπής λεπτομέρειας που αναφέραμε προηγουμένως.



Συνήθως όλα τα μοντέλα που χρησιμοποιούνται αναπτύχθηκαν από κάποιον άνθρωπο και είναι φορτωμένα στην μνήμη ταυτόχρονα για γρήγορη αλλαγή. Ωστόσο, υπάρχουν εργαλεία αυτόματης παραγωγής μοντέλων χαμηλότερου επιπέδου LoD τόσο από την Unity όσο και από την Unreal, ενώ συστήματα όπως το Nanite της Unreal εκτελούν τεχνικές LoD σε πραγματικό χρόνο. Αυτά τα εργαλεία επιχειρούν να συγχωνεύσουν δύο γειτονικά τρίγωνα θέτοντας μία νέα κορυφή. Το πρόβλημα με τέτοια εργαλεία είναι ότι δεν γνωρίζουν την πρόθεση του καλλιτέχνη και μπορούν να αφαιρέσουν λεπτομέρεια η οποία είναι απαραίτητη. Στο Nanite για πράδειγμα της Unreal, η βασική ιδέα είναι ότι εκτελεί συνεχές LoD σε πραγματικό χρόνο. Το επιτυγχάνει αυτό με μία ιεραρχία κορυφών. Κάθε κορυφή σε ένα πλέγμα είναι ένα φύλλο στο διάγραμμα. Μία συστολή ακμής συγχωνεύει τις κορυφές

αδέρφια δημιουργώντας έναν κοινό γονέα. Στη συνέχεια, το κόψιμο του δέντρου καθορίζει ένα σύνολο συστολών που εφαρμόζονται στο πλέγμα. Αν και το Nanite είναι εύκολο στη χρήση του, επειδή πρέπει να γίνεται η προσαρμογή κάθε frame, είναι αρκετά πιο αργό από την χρήση προφορτωμένων μοντέλων. Μέχρι στιγμής έχουμε αναλύσει μόνο LoD στον τομέα του πλέγματος, αλλά γνωρίζουμε ότι τα σημερινά παιχνίδια είναι πολλά περισσότερα από απλά πολύγωνα. Υπάρχει πληροφορία ως προς τον φωτισμό, το χρώμα, τις υφές και άλλα. Το αυτόματο LoD θα πρέπει να λάβει υπόψη και αυτές τις παραμέτρους καθώς κάνει τις συστολές ακμών. Πολλές φορές όμως, η είσοδος από τον καλλιτέχνη είναι απαραίτητη για να βεβαιωθούμε ότι δεν χάνεται σημαντική πληροφορία.



Geometry & texture: A 3,872 face model (a) reduced to 53 faces without (b) and with (c) updating texture coordinates.

Μία Μικρή Υπενθύμιση: Χρώματα και Εικόνες

Πριν προχωρήσουμε στον φωτισμό, είναι καλό να θυμίσουμε λίγο βασικά θέματα που αφορούν το χρώμα και τις εικόνες, αφού στο τέλος στην οθόνη μας προβάλλεται μία ακολουθία από τέτοιες εικόνες. Ένα θέμα αφορά τη χρωματική σύνθεση. Συνήθως χρησιμοποιούμε εικόνες οι οποίες δεν έχουν μόνο χρώμα, αλλά και άλλη πληροφορία όπως το επίπεδο διαφάνειας (Alpha Compositing). Αυτό χρησιμοποιείται για τη σύνθεση εικόνων ημιδιαφανών εικόνων.

Η πιο γνωστή αναπαράσταση χρώματος είναι το προσθετικό τριχρωματικό, γνωστό και ως RGB (Red Green Blue). Χρησιμοποιώντας αυτά τα τρία βασικά χρώματα (ονομάζονται κανάλια) και αναμιγνύοντας τα, μπορούμε να σχεδιάσουμε όποιο χρώμα χρειαζόμαστε. Μπορούμε επίσης να προσθέσουμε το κανάλι διαφάνειας με το μοντέλο RGBa. Ένα άλλο σύστημα είναι το αφαιρετικό χρώμα CMY (Cyan, Magenta, Yello). Αποτελεί το συμπληρωματικό του προσθετικού χρώματος και χρησιμοποιείται κυρίως σε εκτυπωτές καθώς είναι πιο εύκολο να εφαρμοστεί στο λευκό χαρτί.

Κάθε κανάλι χρώματος στο προσθετικό χρώμα έχει μία ένταση, όπου 0 σημαίνει καθόλου φως και 1 σημαίνει μέγιστη ένταση. Αφού μιλάμε για υπολογιστές αυτές οι τιμές συνήθως αναπαρίστανται σε ένα byte (0~255). Οπότε το χρώμα (σε δεκαεξαδικό) FF000080 σημαίνει πλήρης ένταση σε κόκκινο, καθόλου ένταση σε πράσινο και μπλε, και 50% διαφάνεια. Η κβαντοποίηση των τιμών έντασης κάθε καναλιού χρώματος δεν επηρεάζει το τελικό

αποτέλεσμα μιας και το ανθρώπινο μάτι δεν μπορεί να διακρίνει περισσότερα χρώματα από αυτά που μπορούν να αναπαρασταθούν με αυτόν τον τρόπο.

Φωτισμός

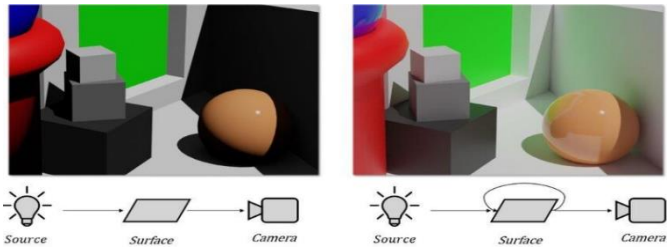
Γυρίζοντας στις τρεις διαστάσεις, ας μιλήσουμε για φωτισμό. Ο φωτισμός είναι συχνά το πιο βασικό στοιχείο ενός σύγχρονου τρισδιάστατου παιχνιδιού, τουλάχιστον ενός που στοχεύει τον ρεαλισμό. Πως υπολογίζουμε όμως το φως; Στα παλαιότερα παιχνίδια όπου η υπολογιστική ισχύ ήταν πολύ χαμηλότερη από ότι σήμερα, μπορεί να μην είχαν πολύπλοκες υλοποιήσεις φωτός, αλλά εφάρμοζαν μία βασική ένταση στα χρώματα των υφών. Σήμερα, για περισσότερο ρεαλισμό λαμβάνοντας υπόψη και την απόδοση, χρησιμοποιούμε πηγές φωτός, ιδιότητες υλικών, και γεωμετρικές σχέσεις αντικειμένων (κυρίως την σχέση του εκάστοτε αντικειμένου με την κάμερα).

The same scene from *The Last of Us: Remastered* (© 2014/™ SIE. Created and developed by Naughty Dog, PlayStation 4) with only diffuse textures applied.

Scene from *The Last of Us: Remastered* (© 2014/™ SIE. Created and developed by Naughty Dog, PlayStation 4) with full lighting.

Η πραγματική αναπαράσταση του φωτός είναι ακριβή υπολογιστικά. Μία ακτίνα από μία πηγή φωτός μπορεί να χτυπήσει το σημείο Α, από το οποίο ανακλάται και χτυπά ένα άλλο σημείο Β. Εντωμεταξύ, στο σημείο Β μπορεί να χτυπάει και το αρχικό φως της πηγής και κάποιο διάσπαρτο φως να χτυπά το σημείο Α. Βλέπουμε πόσο γρήγορα αυξάνονται οι πράξεις που θα πρέπει να κάνουμε για να αναπαραστήσουμε το φως ρεαλιστικά. Τεχνικές όπως το Ray Tracing μπορούν να πλησιάσουν τέτοια επίπεδα ρεαλισμού, αλλά με μεγάλο υπολογιστικό κόστος. Μέσα στον τομέα του φωτισμού είναι φυσικά και το πρόβλημα της σκίασης. Αν δεν επιτρέπαμε στο φως να ανακλαστεί στις επιφάνειες (τοπικός φωτισμός) τότε θα δημιουργούνταν άγριες μαύρες σκιές όπου δεν έφτανε το φως.

Έχουμε επίσης να λύσουμε και το πρόβλημα των επιφανειών. Το φως που προσπίπτει σε ένα αντικείμενο μπορεί να απορροφάται και να σκεδάζεται (ανακλάται) μερικώς ή καθόλου. Η ποσότητα που θα ανακλαστεί καθορίζει το χρώμα και τη φωτεινότητα του αντικειμένου. Το ανακλώμενο φως σκεδάζεται ανάλογα με την ομαλότητα και τον προσανατολισμό της επιφάνειας, άρα μπορεί να σκεδάζεται ομοιόμορφα ή να διασπείρεται ανομοιόμορφα σε πολλές κατευθύνσεις. Ακόμα και η μοντελοποίηση μία πηγής φωτός μπορεί να φανεί προβληματική, αφού μία πηγή δεν είναι ένα σημείο, αλλά ένας όγκος.



Οι τέσσερις πιο βασικές πηγές φωτός που χρησιμοποιούμε είναι οι ακόλουθες:

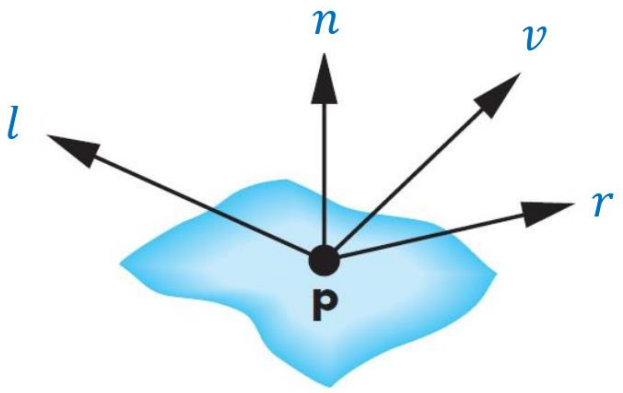
Σημειακή Πηγή: Μπορούμε να φανταστούμε αυτή την πηγή σαν μία λάμπα η οποία φέγγει προς όλες τις κατευθύνσεις. Μοντελοποιείται ως μία σφαίρα με θέση, ακτίνα, και χρώμα φωτός. Αν θέσουμε μία τέτοια πηγή σε θέση με άπειρη απόσταση από τον χώρο μας, τότε οι ακτίνες του φωτός που φτάνουν στον χώρο μας είναι πρακτικά παράλληλες και η πηγή λειτουργεί πλέον ως μία μακρινή πηγή, σαν τον ήλιο.

Προβολέας: Μπορούμε να φανταστούμε αυτή την πηγή σαν έναν φακό. Μοντελοποιείται σαν μία σημειακή πηγή η οποία ακτινοβολεί σε μία συγκεκριμένη γωνία, ώστε πρακτικά ο χώρος να είναι σε σχήμα κώνου. Μία τέτοια πηγή χρειάζεται θέση, ακτίνα, και χρώμα όπως η σημειακή πηγή, αλλά απαιτεί επίσης και την γωνία περιορισμού του φωτός, καθώς και την κατεύθυνση στην οποία δείχνει ο κώνος.

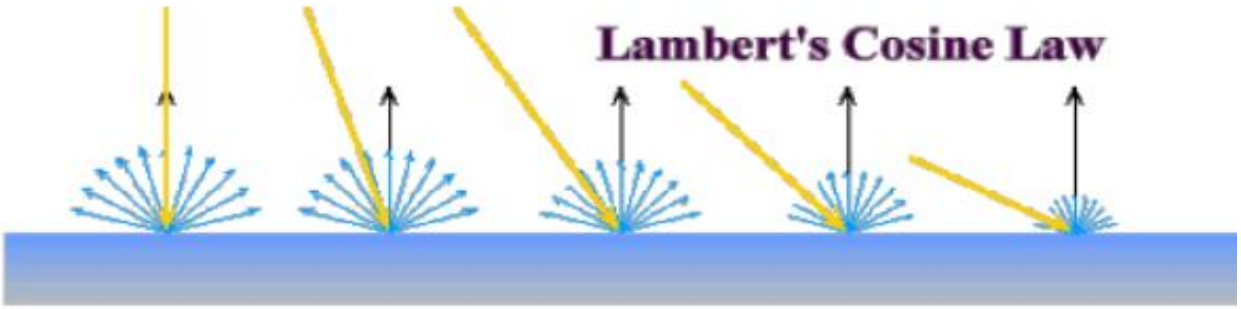
Πηγή Περιβάλλοντος: Μάλλον η πιο απλή πηγή φωτός. Εφαρμόζει την ίδια ποσότητα φωτός σε όλη τη σκηνή. Για τη μοντελοποίηση αυτής της πηγής απαιτείται μόνο το χρώμα του φωτός καθώς και κάποιες άλλες ρυθμίσεις όσον αφορά την αλληλεπίδρασή του με ανακλαστικές επιφάνειες. Χρησιμοποιείται συνήθως για την εφαρμογή κάποιου χαμηλού ορίου στην σκηνή, ώστε να μην υπάρχει σημείο το οποίο να είναι απολύτως μαύρο.

Όσον αφορά το πρόβλημα της ανάκλασης σε ένα σημείο εφαρμόζουμε το Μοντέλο του Phong. Είναι ένα απλό μοντέλο το οποίο υπολογίζεται εύκολα αν και απέχει πολύ από την πραγματικότητα. Ωστόσο, παρουσιάζει ικανοποιητικά αποτελέσματα. Το χρώμα σε ένα σημείο, δηλαδή μία κορυφή, υπολογίζεται από τρεις συνιστώσες, την διάχυτη ανάκλαση, την κατοπτρική ανάκλαση, και την ανάκλαση από το περιβάλλον. Χρησιμοποιούνται τέσσερα διανύσματα με αρχή το σημείο $\boldsymbol{p}$. Ένα διάνυσμα από το $\boldsymbol{p}$ προς την κάμερα $\boldsymbol{v}$, ένα από το σημείο προς την πηγή φωτός $\boldsymbol{l}$, ένα κάθετο διάνυσμα στο σημείο $\boldsymbol{n}$, και το διάνυσμα που δείχνει την τέλεια ανάκλαση από την πηγή του φωτός $\boldsymbol{r}$.

Επιπροσθέτως, εφαρμόζουμε το φως περιβάλλοντος. Όπως αναφέραμε προηγουμένως, αποτελεί το φως που προκύπτει από τις αντανακλάσεις άλλων επιφανειών στη σκηνή. Επειδή είναι αδύνατον να το υπολογίσουμε σε πραγματικό χρόνο, το μοντελοποιούμε σαν ένα γενικό επίπεδο φωτεινότητας το οποίο εφαρμόζεται σε όλη την σκηνή και είναι ανεξάρτητο από τον προσανατολισμό της κάθε επιφάνειας. Κάθε πηγή φωτός έχει μία συμβολή φωτός περιβάλλοντος L_a. Για κάθε επιφάνεια καθορίζουμε πόσο φως περιβάλλοντος μπορεί να αντανακλά η επιφάνεια χρησιμοποιώντας το συντελεστή ανάκλασης περιβάλλοντος $0 \leq K_a \leq 1$. Τελικά μία επιφάνεια αντανακλά $I_a = L_a \cdot K_a$.



Ένα άλλο μέρος της μοντελοποίησης του φωτός είναι το διάχυτο φως. Αποτελεί τον φωτισμό που λαμβάνει μία επιφάνεια από μία πηγή και αντανακλά εξίσου ομοιόμορφα προς όλες τις κατευθύνσεις. Αυτές οι επιφάνειες ονομάζονται Lambertian και η φωτεινότητα της επιφάνειας εξαρτάται μόνο από την γωνία μεταξύ του κάθετου διανύσματος της επιφάνειας και της πηγής φωτός. Υπολογίζεται από τον τύπο $L_d \cdot \cos q$ όπου L_d είναι η συνιστώσα διάχυσης πηγής και q είναι η προαναφερθείσα γωνία. Σε αυτό το βήμα εφαρμόζεται και ένας συντελεστής ανάκλασης διάχυσης $0 \leq K_d \leq 1$, οπότε το διάχυτο φως που αντανακλάται από ένα σημείο της επιφάνειας είναι $I_d = K_d \cdot L_d \cdot \cos q$.



Μπορεί να έχετε παρατηρήσει σε σφαιρικές λείες επιφάνειες, όπως ένα μπαλόνι, ένα σημείο στο οποίο το φως ανακλά τέλεια ώστε να είναι πολύ πιο φωτεινό από την υπόλοιπη επιφάνεια. Αυτό ονομάζεται κατοπτρικό φως και το αποδίδουμε μοντελοποιώντας την ιδιότητα επιφάνειας του υλικού. Έστω ότι η επιφάνεια μπορεί να ανακλά $0 \leq K_d \leq 1$ ποσοστό φωτός κατοπτρικής ανάκλασης. Τότε το συνολικό κατοπτρικό φως σε ένα σημείο

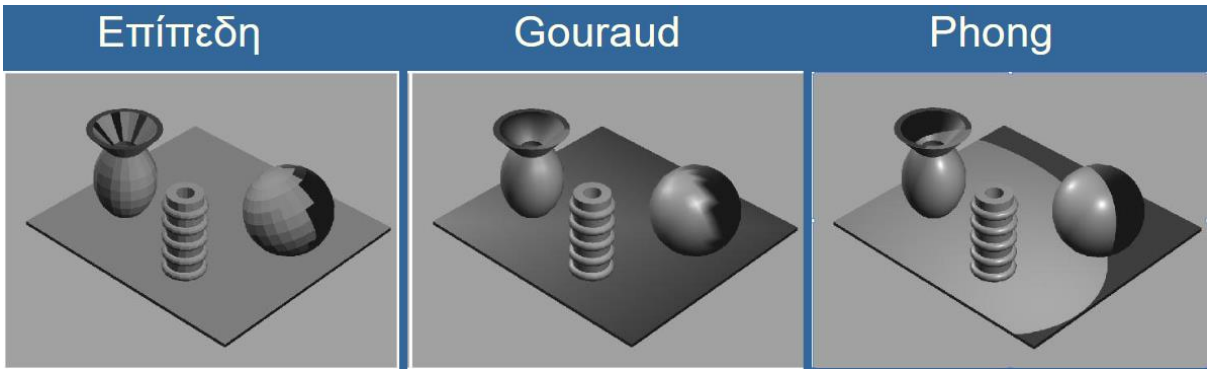
είναι $I_s = K_s \cdot L_s \cdot \cos^{\alpha} \varphi$, όπου το α δηλώνει την μεταλλότητα της επιφάνειας, με τιμές μεταξύ 100 και 200 να αντιστοιχούν σε μεταλλικές επιφάνειες, ενώ τιμές μεταξύ 5 και 10 ή μικρότερες να αντιστοιχούν σε πιο πλαστικές επιφάνειες.

Συνδυάζοντας όλες τις παραπάνω συνιστώσες, το σημειακό φως είναι το άθροισμα του φωτός του περιβάλλοντος με το διάχυτο φως και το κατοπτρικό φως. Άρα:

$$I = I_a + I_d + I_s = K_a L_a + K_d L_d \cos q + K_s \cdot L_s \cdot \cos^{\alpha} \varphi$$

Αν έχουμε περισσότερες από μία πηγές φωτός απλώς προσθέτουμε τις συνεισφορές κάθε πηγής.

Ο υπολογισμός φωτισμού εφαρμόζεται συνήθως σε συγκεκριμένα σημεία (π.χ., στις κορυφές του τριγώνου). Ο φωτισμός των υπολοίπων σημείων του τριγώνου γίνεται μέσω διαδικασιών που περιγράφονται συνολικά ως σκίαση (shading). Πολλές φορές υπάρχει σύγχυση των όρων της σκίασης με τη δημιουργία των σκιών που είναι όμως διαφορετικά. Τα πιο βασικά μοντέλα σκίασης για πολύγωνα είναι τα ακόλουθα:



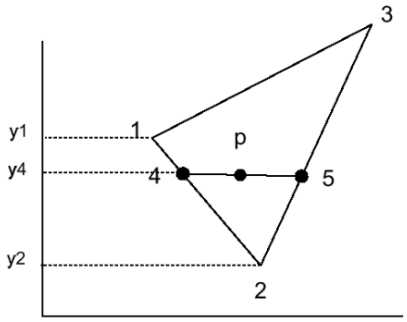
Σταθερή ή επίπεδη Σκίαση: Υπολογίζουμε τον φωτισμό σε ένα σημείο της επιφάνειας χρησιμοποιώντας το κάθετο διάνυσμα της επιφάνειας του τριγώνου. Τα υπόλοιπα σημεία του τριγώνου απλά αντιγράφουν το συγκεκριμένο θέμα. Είναι ικανοποιητικό για μακρινό φως και μακρινό θεατή ή εάν οι όψεις προσεγγίζουν καλά την επιφάνεια, δηλαδή αν το αντικείμενο δεν έχει καμπυλωτή μορφή.

Σκίαση με Παρεμβολή: Υπολογίζουμε τον φωτισμό στις κορυφές και εφαρμόζουμε παρεμβολή στα άλλα σημεία. Τέτοιο μοντέλο σκίασης είναι η σκίαση Gouraud σύμφωνα με την οποία το χρώμα παρεμβάλλεται σε όλη την επιφάνεια του τριγώνου. Η τεχνική αυτή είναι γρήγορη αλλά προκαλεί άσχημα αποτελέσματα αν εμφανιστεί κατοπτρικό φως στο κέντρο ενός πολυγώνου.

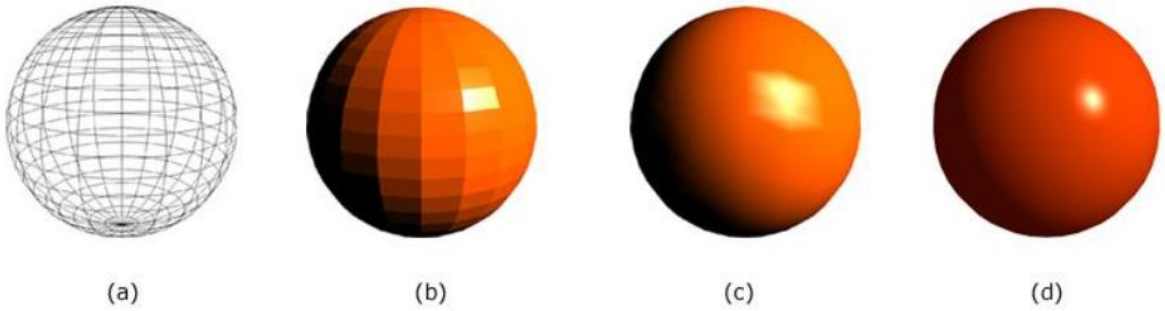
Σκίαση ανά Εικονοστοιχείο: Υπολογίζουμε το φωτισμό για κάθε σημείο της επιφάνειας. Με αυτό το μοντέλο παρεμβάλλονται κάθετα διανύσματα στην επιφάνεια του τριγώνου. Φέρνει

καλύτερα αποτελέσματα από την σκίαση με παρεμβολή και χειρίζεται πολύ καλύτερα το κατοπτρικό φως, αλλά είναι πιο ακριβό υπολογιστικά.

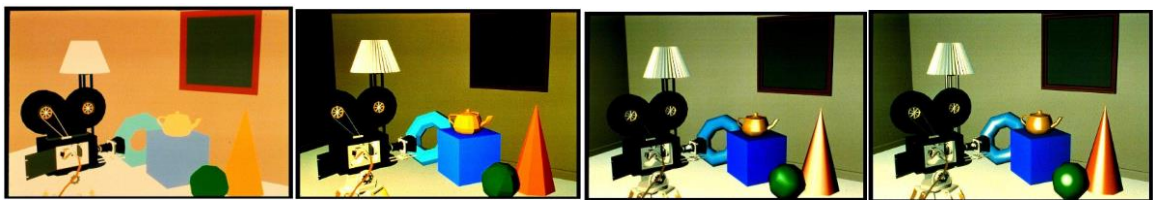
Στο διπλανό σχήμα φαίνεται η Gouraud σκίαση σε ένα τρίγωνο. Για να προσδιορίσουμε την ένταση του σημείου p στο τρίγωνο ακολουθούμε τα παρακάτω βήματα. Πρώτα προσδιορίζουμε την ένταση των σημείων 4 και 5 παρεμβάλλοντας γραμμικά μεταξύ των 1 και 2, και 2 και 3 αντίστοιχα. Στη συνέχεια προσδιορίζουμε την ένταση του p με γραμμική παρεμβολή μεταξύ των 4 και 5.



Είναι πιο δύσκολο να κάνουμε παράδειγμα με την σκίαση Phong αλλά τα βασικά σημεία είναι τα εξής: Πρώτα υπολογίζουμε τα κάθετα διανύσματα των κορυφών. Στη συνέχεια, εφαρμόζουμε παρεμβολή των κάθετων διανυσμάτων κατά μήκος των ακμών και των εσωτερικών σημείων. Τέλος, υπολογίζουμε τον φωτισμό ανά pixel. Στο παρακάτω σχήμα φαίνεται η εφαρμογή επίπεδης (b), Gouraud (c) και Phong (d) στο μοντέλο (a).



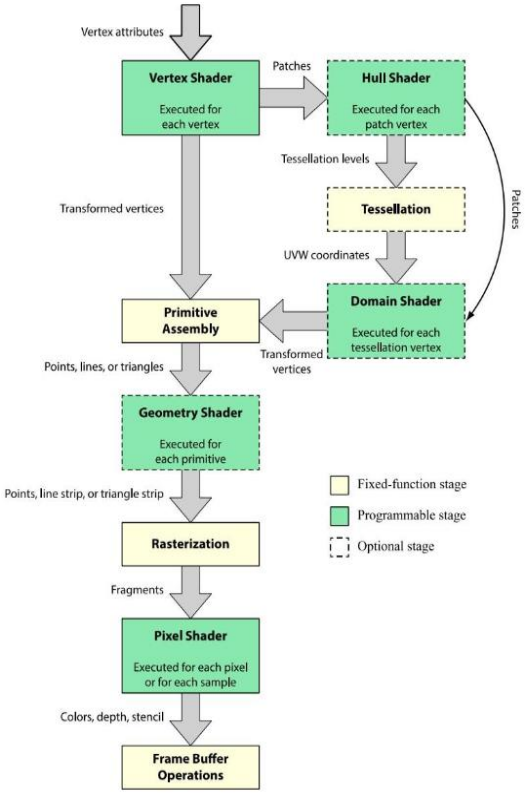
Ομοίως, στην παρακάτω σκηνή η πρώτη εικόνα δεν έχει καθόλου σκίαση, η δεύτερη εφαρμόζει επίπεδη σκίαση, η τρίτη σκίαση Gouraud, και η τέταρτη σκίαση Phong.



Προγραμματιζόμενοι Shaders

Οι σύγχρονες κάρτες γραφικών εισήγαγαν την ιδέα προγραμμάτων ονόματι shaders. Η αρχική τους λειτουργία αφορούσε τον φωτισμό και την σκίαση επιφανειών, αλλά σήμερα η

λέξη shader σημαίνει οποιαδήποτε σειρά προγραμματιζόμενων μονάδων GPU. Αν δηλαδή θέλουμε να τρέξουμε ένα γενικό πρόγραμμα σε μία GPU θα το γράψουμε σαν shader. Στη διπλανή εικόνα φαίνεται μία από τις πολλές αρχιτεκτονικές shader. Στις μηχανές παιχνιδιών δεν είναι απαραίτητο να χρησιμοποιούνται όλοι αυτοί οι shaders. Επίσης, δεν υποστηρίζουν όλα τα API απόδοσης γραφικών όλους αυτούς τους τύπους shader. Εκτός από αυτούς τους shaders, υπάρχουν και άλλοι τύποι, όπως: Tessellation Shaders, Mesh Shaders, Ray-Tracing Shaders, Compute Shaders.



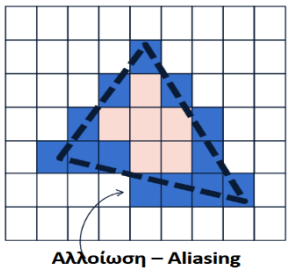
Shader Κορυφών: Κάθε κορυφή στη γεωμετρία της σκηνής τίθεται προς επεξεργασία από ένα πρόγραμμα σκίασης κορυφής. Συνήθως εφαρμόζουμε μόνο γεωμετρικούς μετασχηματισμούς που σχετίζονται με animation, προβολή, και άλλα. Το πρόγραμμα εκτελείται πανομοιότυπα σε πολλούς πυρήνες κορυφών της GPU. Μία γνωστή εφαρμογή αυτού του shader είναι η προσομοίωση κυμάτων στη θάλασσα. Μπορούμε δηλαδή να φτιάξουμε ένα μεγάλο επίπεδο με πολλές κορυφές και να εφαρμόσουμε μία πάνω-κάτω κίνηση σε κάθε κορυφή η οποία προσομοιάζει τους σχηματισμούς των κυμάτων στον ωκεανό.

Pixel Shader: Εκτελείται για κάθε pixel στην οθόνη αφού έχει ολοκληρωθεί η απόδοση γραφικών και η ραστεροποίηση. Συνήθως υπολογίζει ένα χρώμα για αυτό το pixel. Το πρόγραμμα εκτελείται πανομοιότυπα σε πολλούς πυρήνες κορυφών της GPU. Μία απλή εφαρμογή αυτού του shader είναι να αλλάζει το χρώμα ενός χαρακτήρα (συνήθως σε λευκό) όταν έχει χτυπηθεί για να συμβολίσει ότι είναι σε κατάσταση όπου δεν μπορεί να τραυματιστεί άλλο.

Ραστεροποίηση

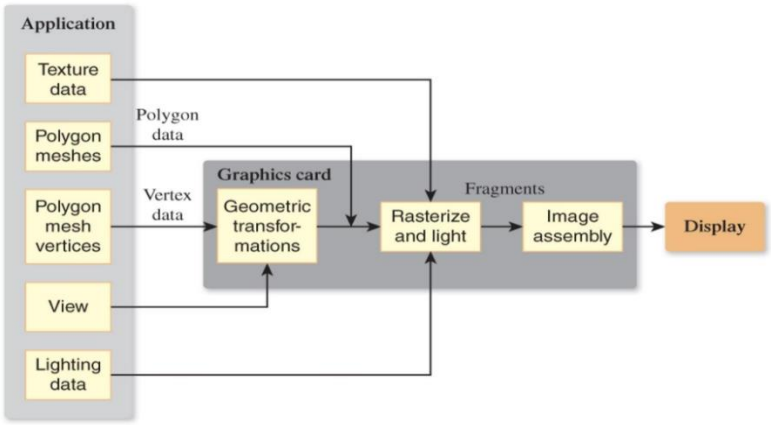
Έχουμε ξαναμιλήσει για τον ραστεροποιητή στη διάλεξη σχετικά με τις κάμερες, αλλά μόνο για την προβολή του τρισδιάστατου χώρου στην οθόνη. Στην πραγματικότητα, ο ραστεροποιητής κάνει πολλά περισσότερα από αυτό. Για παράδειγμα, ο ραστεροποιητής είναι υπεύθυνος για την εφαρμογή της υφής στα τρίγωνα ή αποκόπτει τρίγωνα που δεν εμφανίζονται με χρήση της τεχνικής z-buffering.

Ως είσοδο δέχεται τις κορυφές των τριγώνων από το στάδιο της γεωμετρίας. Η επεξεργασία αυτής της πληροφορίας, έχει ως αποτέλεσμα την ανίχνευση των pixel μέσα στο τρίγωνο (ή σε μία γραμμή ή σε ένα σημείο). Στη συνέχεια, εκτελεί λειτουργίες ανά pixel, όπως παρεμβολή, εφαρμογή υφών, Z-buffering, και άλλα.


Αλλοίωση – Aliasing

Ένα pixel (εικονοστοιχείο) είναι το μικρότερο ελεγχόμενο στοιχείο μίας εικόνας. Ο συνδυασμός χιλιάδων pixel με διαφορετικά χρώματα δημιουργεί μία εικόνα. Ένα ράστερ είναι ένα πλέγμα τιμών εικονοστοιχείων. Τυπικά είναι ένα ορθογώνιο πλέγμα τιμών χρώματος. Η ραστεροποίηση είναι η διαδικασία κατά την οποία δημιουργείται ένα ράστερ εικόνας από μία διανυσματική περιγραφή. Επειδή είναι αδύνατον να αποδώσουμε τέλεια ένα πολύγωνο στην οθόνη παρουσιάζεται αλλοιώσεις που μπορούν να διορθωθούν με antialiasing μεθόδους.

Θυμίζουμε ότι η ροή εργασίας για 3d γραφικά ξεκινά με την επεξεργασία κορυφών, ακολουθεί η ραστεροποίηση όπου παράγονται και τα θραύσματα (fragments) των τριγώνων. Τα θραύσματα αντιπροσωπεύουν μία μικρή περιοχή της επιφάνειας του τριγώνου που αντιστοιχεί σε ένα μόνο pixel στην οθόνη. Στην περίπτωση της τεχνικής multisample antialiasing ένα θραύσμα αντιστοιχεί σε ένα τμήμα ενός pixel.



Όπως είπαμε, η ραστεροποίηση παράγει ένα σύνολο θραυσμάτων για κάθε τρίγωνο. Τα θραύσματα είναι εν δυνάμει pixels τα οποία αποθηκεύονται στον frame buffer και έχουν ιδιότητες χρώματος και βάθους. Ο fragment shader εφαρμόζει παρεμβολή στα χαρακτηριστικά των κορυφών μεταξύ των θραυσμάτων. Να σημειωθεί ότι εφαρμόζεται σε κάθε θραύσμα ξεχωριστά όπως ήταν με τα αρχικά του χαρακτηριστικά χωρίς να λαμβάνει υπόψιν άλλες τυχόν αλλαγές που μπορεί να εφάρμοσε.

Buffering

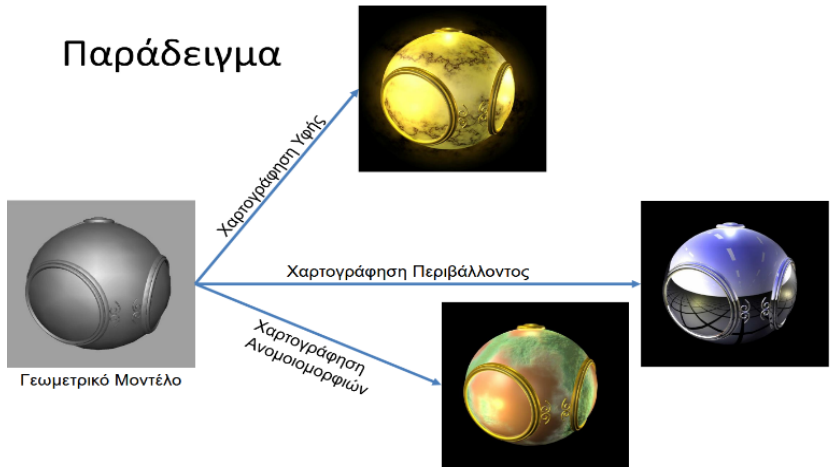
Όταν ολοκληρωθεί η εικόνα, αποθηκεύεται στον frame buffer. Το υλικό απεικόνισης σαρώνει τον frame buffer και απεικονίζει τα περιεχόμενά του. Συνήθως υπάρχουν δύο frame buffers, ώστε όσο ο ένας σαρώνει, ό δεύτερος ήδη να ετοιμάζει την επόμενη εικόνα. Η τεχνική αυτή ονομάζεται double buffering.

Χαρτογράφηση

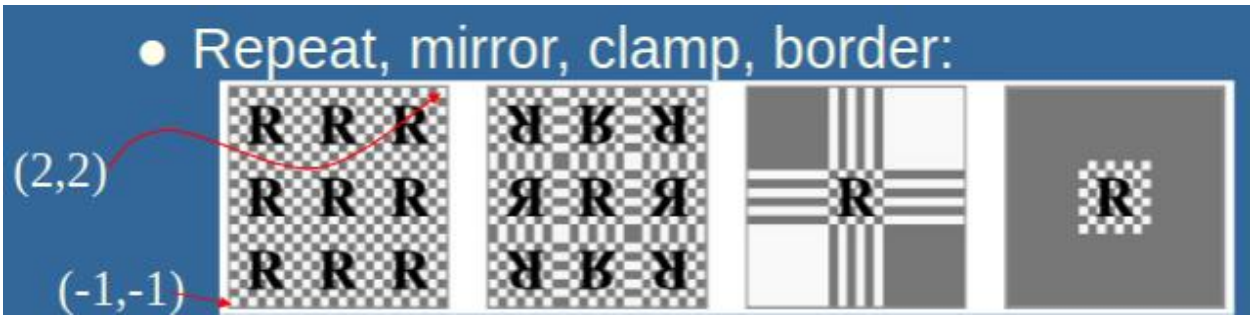
Υπάρχουν όρια στην γεωμετρική μοντελοποίηση. Παρόλο που οι σύγχρονες κάρτες γραφικών είναι ικανές να αποδώσουν δεκάδες εκατομμύρια πολύγωνα πάνω από 60 φορές το δευτερόλεπτο, δεν αρκούν για φαινόμενα όπως τα σύννεφα, το γρασίδι, το έδαφος και το δέρμα, στα οποία υπάρχει πολύ λεπτομέρεια που χαρακτηρίζει την ιδιαιτερότητά τους.

Ας πάρουμε για παράδειγμα ένα πορτοκάλι. Θα μπορούσαμε να απεικονίσουμε το πορτοκάλι σαν μία σφαίρα με πορτοκαλί χρώμα, αλλά το αποτέλεσμα θα ήταν πολύ φτωχό και δεν αντιστοιχεί στο πραγματικό σχήμα του φρούτου. Αντί για μία σφαίρα, μπορούμε να φτιάξουμε ένα πιο πολύπλοκο σχήμα, αλλά πάλι θα χάνονται οι ανομοιομορφίες και οι ατέλειες της επιφάνειας του πορτοκαλιού, εκτός αν χρησιμοποιήσουμε υπερβολικά πολλά πολύγωνα για να το μοντελοποιήσουμε ορθά, το οποίο υπολογιστικά είναι ακριβό. Εναλλακτικά, θα μπορούσαμε να χρησιμοποιήσουμε μία φωτογραφία ενός πορτοκαλιού και να την επικολλήσουμε σε ένα απλό γεωμετρικό μοντέλο. Αυτή η διαδικασία είναι γνωστή ως χαρτογράφηση υφής. Η επιφάνεια που προκύπτει, ωστόσο, θα είναι λεία, άρα απαιτείται η αλλαγή του τοπικού σχήματος εφαρμόζοντας χαρτογράφηση ανομοιομορφιών.

Υπάρχουν διάφοροι τύποι χαρτογράφησης. Κατά την χαρτογράφηση υφής (texture mapping) χρησιμοποιούμε εικόνες για να γεμίσουμε το εσωτερικό των πολυγώνων. Κατά την χαρτογράφηση περιβάλλοντος (environment mapping) χρησιμοποιούμε εικόνες του περιβάλλοντος ως υφή, ενώ προσομοιώνουμε ανακλαστικές επιφάνειες. Κατά την χαρτογράφηση ανομοιομορφιών (bump mapping) εξομοιώνουμε τα διαφορετικά κάθετα διανύσματα με στόχο να μοντελοποιήσουμε μικρές ανομοιομορφίες στην επιφάνεια. Οι τεχνικές αυτές εφαρμόζονται στο τέλος της απόδοσης γραφικών κατά την επεξεργασία θραυσμάτων.



Προκειμένου να δώσουμε χρώμα στα μοντέλα, εφαρμόζουμε πάνω τους μία εικόνα υφής (texture). Ο χώρος της υφής, δηλαδή το σύστημα συντεταγμένων της, είναι συνήθως κανονικοποιημένος στο διάστημα [0,1]. Η βασική ιδέα είναι απλή, αλλά στην πράξη εμπλέκονται 3 ή και παραπάνω συστήματα συντεταγμένων. Έχουμε το αρχικό σύστημα της εικόνας, το σύστημα συντεταγμένων του τρισδιάστατου χώρου, το σύστημα συντεταγμένων του παραθύρου της κάμερας, και το σύστημα συντεταγμένων της οθόνης. Αν κατά τη μετατροπή των συστημάτων συντεταγμένων κάποιες βρεθούν εκτός ορίων της υφής χρησιμοποιούμε διάφορες μεθόδους. Μπορούμε να επαναλάβουμε την υφή, να την καθρεπτίσουμε και μετά να την επαναλάβουμε, να χρησιμοποιήσουμε μόνο το πιο ακριανό χρώμα της υφής, ή να περιοριστούμε στη χρήση μόνο της διαθέσιμης εικόνας, αφήνοντας την επιφάνεια που δεν καλύπτεται διάφανη, ή με ένα βασικό χρώμα.



Μία τεχνική LoD η οποία εφαρμόζεται στις υφές είναι η Mipmapping. Κάθε στοιχείο μία υφής ονομάζεται texel. Κατά τη διάρκεια χαρτογράφησης υφής πολλά texels μπορεί να αντιστοιχούν σε ένα pixel. Για σωστή αντιστοίχιση, παίρνουμε τον μέσο όρο από όλα αυτά τα texel, το οποίο είναι ακριβό υπολογιστικά. Για να αποφύγουμε αυτό το πρόβλημα, προϋπολογίζουμε τις υφές σε διαφορετικές αναλύσεις, ιδανικά σε δυνάμεις του 2. Τέλος, υπολογίζουμε το επίπεδο λεπτομέρειας από υπολογισμό texel ανά pixel. Σαν αποτέλεσμα, χρειαζόμαστε περίπου 33% περισσότερο χώρο στη μνήμη για την αποθήκευση και των

υφών μικρότερης ανάλυσης. Από την άλλη πλευρά, η απόδοση είναι καλύτερη, ιδιαίτερα για απομακρυσμένα αντικείμενα.

Πριν το ray tracing, χρειαζόμασταν διαφορετικές τεχνικές για την αναπαράσταση επιφανειών υψηλής ανακλαστικότητας. Μία σημαντική τεχνική είναι η χαρτογράφηση του περιβάλλοντος. Αποτελεί μία εικόνα του περιβάλλοντος μέσα από έναν ευρυγώνιο φακό, φανταστείτε το σαν την εικόνα μίας 360^o κάμερα. Χρησιμοποιούμε την παραγόμενη υφή για να δημιουργήσουμε έναν – συνήθως – σφαιρικό ή κυβικό χάρτη. Για τη δημιουργία ενός χάρτη κύβου θα χρειαζόμασταν 6 κάμερες με 90^o μοίρες γωνία θέασης η κάθε μία. Για την χαρτογράφηση απαιτείται ο υπολογισμός του διανύσματος ανάκλασης r. Έχοντας υπολογίσει την υφή μας μπορούμε να την εφαρμόσουμε σε μία ανακλαστική επιφάνεια, όπως ένας καθρέφτης. Το πρόβλημα με αυτή την τεχνική είναι ότι πρέπει να υποθέσουμε ότι το περιβάλλον είναι αρκετά μακριά από το αντικείμενο, έτσι ώστε να μην επηρεάζεται πολύ η τελική υφή από την γωνία θέασης (όπως και ο θεατής). Επιπροσθέτως, δεν υποστηρίζονται οι αυτοανακλάσεις ή οι ανακλάσεις μεταξύ άλλων ανακλαστικών αντικειμένων. Τέλος, θα πρέπει να δημιουργήσουμε καινούριο χάρτη όχι μόνο για κάθε καινούργιο αντικείμενο, αλλά και για κάθε φορά που το αντικείμενο κινείται. Η τεχνική αυτή είναι περισσότερο αποτελεσματική για μακρινά ακίνητα/στατικά αντικείμενα, ωστόσο τίποτα δεν μας εμποδίζει να τη χρησιμοποιούμε σε συνδυασμό με ανίχνευση ανάκλασης πραγματικού χρόνου για το κοντινό περιβάλλον, όπως τον κινούμενο παίκτη.

Μία άλλη μορφή χαρτογράφησης είναι η χαρτογράφηση ανομοιομορφιών. Αποτελεί έναν φθηνό τρόπο να προσομοιώσουμε μικρές ανομοιομορφίες σε μία γεωμετρία, όπως οι ρυτίδες στο δέρμα. Μία υφή ανομοιομορφίας αλλάζει το κάθετο διάνυσμα σε κάθε pixel και έπειτα χρησιμοποιεί το παραγόμενο διάνυσμα για να υπολογίσει τον φωτισμό. Για την υλοποίηση, χρησιμοποιείται ένας χάρτης υφής σε κλίμακα του γκρι όπου οι σκοτεινές περιοχές αντιπροσωπεύουν τις χαμηλότερες περιοχές ενώ οι φωτεινές περιοχές αντιπροσωπεύουν τις υψηλότερες περιοχές. Ορίζουμε το κάτω και το άνω όριο της κλίμακάς μας και το κανονικοποιούμε χρησιμοποιώντας την αντίστοιχη τιμή του χάρτη σαν ποσοστό. Για παράδειγμα, ας υποθέσουμε ότι έχουμε ένα πλέγμα σε σχήμα τετραγώνου με 101 επί 101 κορυφές. Επίσης έχουμε τον χάρτη ανομοιμορφιών ο οποίος είναι 101 επί 101 pixels. Ορίζουμε κάτω όριο -5 μονάδες και άνω όριο +5 μονάδες. Αν το ακριβώς μεσαίο pixel της οθόνης είναι εντελώς μαύρο, τότε η ακριβώς μεσαία κορυφή θα μετακινηθεί 5 μονάδες προς τα κάτω. Στην πραγματικότητα δεν μετακινούμε τις κορυφές, αλλά τροποποιούμε τα διανύσματα της επιφάνειας ώστε να εμφανιστεί η ψευδολεπτομέρεια κατά τον υπολογισμό της σκίασης. Το αποτέλεσμα είναι πανομοιότυπο με το να μετακινούσαμε τις κορυφές, ή να προσθέταμε περισσότερες, αλλά πολύ φθηνότερο υπολογιστικά.

Τελευταία μορφή χαρτογράφησης για την οποία θα μιλήσουμε είναι η χαρτογράφηση καθετότητας, η οποία χρησιμοποιείται σε εικόνες για να δώσει έναν 3D χαρακτήρα και αποτελεί εξέλιξη της χαρτογράφησης ανομοιομορφίας. Σε αντίθεση με τη χαρτογράφηση ανομοιομορφίας όπου βρίσκουμε το κάθετο διάνυσμα από το αποθηκευμένο υψόμετρο, εδώ το κάθετο διάνυσμα αποθηκεύεται απευθείας σε μία υφή. Αντί για κλίμακα του γκρι, η υφή μας πλέον χρησιμοποιεί όλα τα χρώματα για να κωδικοποιήσει κανονικά διανύσματα επιφάνειας για κάθε σημείο της. Για παράδειγμα, ένα λευκό χρώμα δηλώνει διάνυσμα κάθετο στην υφή, το πιο μωβ χρώμα δηλώνει διάνυσμα προς τα δεξιά, και το πιο πράσινο χρώμα δηλώνει διάνυσμα προς τα αριστερά. Τελικά, αποθηκεύουμε απευθείας τα κανονικά διανύσματα τα οποία είναι έτοιμα να χρησιμοποιηθούν σε υπολογισμούς σκίασης. Γλιτώνουμε υπολογιστική ισχύ για περισσότερο χώρο. Το αποτέλεσμα όμως είναι πολύ πιο λεπτομερή εφέ φωτισμού, τα οποία συνεισφέρουν στον ρεαλισμό της σκηνής μας.