Εθνικό Μετσόβιο Πολυτεχνείο

Σχολή Αγρονόμων και Τοπογράφων Μηχανικών-Μηχανικών Γεωπληροφορικής

Διπλωματική Εργασία

Απομακρυσμένη απόδοση (Remote Rendering) για φορετές συσκευές σε εφαρμογή VR

Βασίλειος Ν. Λευκαδίτης
Α.Μ.: Rs15014

Επιβλέπουσα καθηγήτρια: Μαρία Πατεράκη

Αθήνα

Ιούλιος 2023

Ευχαριστίες

Η εκπόνηση της παρούσας διπλωματικής εργασίας πραγματοποιήθηκε κατά το ακαδημαϊκό έτος 2022-2023, στο Εργαστήριο Φωτογραμμετρίας του Τομέα Τοπογραφίας της Σχολής Αγρονόμων και Τοπογράφων Μηχανικών-Μηχανικών Γεωπληροφορικής, του Εθνικού Μετσόβιου Πολυτεχνείου.

Αρχικά θα ήθελα να ευχαριστήσω την επιβλέπουσα καθηγήτρια μου Μαρία Πατεράκη για την καθοδήγηση και την ουσιαστική βοήθεια που μου παρείχε στο πλαίσιο της συνεργασίας μας, καθώς και για τις γνώσεις που αποκόμισα από εκείνην, καθ' όλη τη διάρκεια των προπτυχιακών μου σπουδών.

Ακόμα θα ήθελα να ευχαριστήσω τους φίλους μου για τις όμορφες στιγμές και τις μοναδικές αναμνήσεις που δημιουργήσαμε μαζί.

Πάνω από όλους, θα ήθελα να ευχαριστήσω τους γονείς μου, Νικόλαο και Λαμπρινή, αλλά και τον αδερφό μου Γεράσιμο για την στήριξη και την βοήθεια που μου παρείχαν καθ' όλη την διάρκεια των σπουδών μου.

Τέλος θα ήθελα να ευχαριστήσω τα παρακάτω άτομα για την βοήθεια τους στην εκτέλεση της διπλωματικής εργασίας μου: Αγγελική Ζ., Αφροδίτη Κ., Γεράσιμος Λ., Ευγενία Κ., Κωνσταντίνος Ν.

Περιεχόμενα

Περίληψη

Η χρήση ασύρματων συστημάτων εικονικής πραγματικότητας είναι εξαιρετικά διαδεδομένη και κατέχει ύψιστη σημασία σε διάφορους τομείς. Τα συστήματα αυτά προσφέρουν στους χρήστες τη δελεαστική προοπτική της ελευθερίας κινήσεων, εξαλείφοντας έτσι τους περιορισμούς που επιβάλλονται από τα ενσύρματα μέσα. Ωστόσο, μια σημαντική πρόκληση που αντιμετωπίζουν τα HMD είναι η περιορισμένη υπολογιστική τους ισχύς. Για να ξεπεραστεί το συγκεκριμένο εμπόδιο, η εφαρμογή της απομακρυσμένης απόδοσης έχει αναδειχθεί ως βιώσιμη λύση. Ως εκ τούτου, ο πρωταρχικός στόχος αυτής της διατριβής περιστρέφεται γύρω από τη διερεύνηση και την ανάλυση μεθοδολογιών απομακρυσμένης απόδοσης εντός ασύρματων συστημάτων εικονικής πραγματικότητας. Για να ξεκινήσει αυτή η ολοκληρωμένη μελέτη, το πρώτο στάδιο περιλαμβάνει μια εκτενή συλλογή πληροφοριών που περιλαμβάνει διάφορες πτυχές όπως μεθόδους, προκλήσεις, τεχνικές μετάδοσης δεδομένων, στρατηγικές βελτιστοποίησης και αξιολόγησης της απομακρυσμένης απόδοσης. Αυτή η σχολαστική εξέταση στοχεύει να θέσει μια γερή βάση για την κατανόηση των περιπλοκών που σχετίζονται με την απομακρυσμένη απόδοση. Επιπλέον, η μελέτη εμβαθύνει στην ανάλυση δύο ξεχωριστών συστημάτων απομακρυσμένης απόδοσης που χρησιμοποιούνται στην ασύρματη εικονική πραγματικότητα. Με την ενδελεχή εξέταση αυτών των συστημάτων, ο στόχος είναι να αποκτηθούν πολύτιμες γνώσεις για την αποτελεσματική αντιμετώπιση των περιορισμών που θέτουν τα ασύρματα HMD και τη δημιουργία εφαρμογών που διαθέτουν ανώτερη ποιότητα γραφικών. Αυτή η ανάλυση προσφέρει πολύτιμες κατευθυντήριες γραμμές και πιθανές οδούς για την ανάπτυξη εφαρμογών αιχμής στον τομέα της ασύρματης εικονικής πραγματικότητας. Εκτός από τη θεωρητική διερεύνηση, αυτή η διατριβή περιλαμβάνει επίσης την περιγραφή διαφόρων εργαλείων που μπορούν να αξιοποιηθούν για την επίτευξη αποτελεσματικής απομακρυσμένης απόδοσης. Αυτά τα εργαλεία χρησιμεύουν ως απαραίτητοι πόροι για την εφαρμογή και τη βελτιστοποίηση μεθοδολογιών απομακρυσμένης απόδοσης, παρέχοντας σε ερευνητές και προγραμματιστές τα μέσα για να βελτιώσουν τη συνολική εμπειρία VR. Τέλος, για την προβολή των πρακτικών επιπτώσεων των προαναφερθέντων εργαλείων και μεθοδολογιών, αναπτύσσεται μια εφαρμογή. Αυτή η πρακτική εφαρμογή χρησιμεύει ως απόδειξη της αποτελεσματικότητας και των δυνατοτήτων της απομακρυσμένης απόδοσης σε ασύρματα συστήματα εικονικής πραγματικότητας. Με βάση τα ευρήματα και τις παρατηρήσεις που προκύπτουν από την ανάπτυξη της εφαρμογής, εξάγονται διορατικά συμπεράσματα και διατυπώνονται προτάσεις που προκαλούν προβληματισμό για να καθοδηγήσουν μελλοντικές ερευνητικές προσπάθειες σε αυτόν τον τομέα.

Abstract

The utilization of wireless virtual reality systems is remarkably widespread and holds paramount importance in various fields. These systems offer users the enticing prospect of freedom of movement, thereby eliminating the constraints imposed by wired media. However, one prominent challenge faced by wireless head-mounted displays (HMDs) is their limited computing power. To overcome this specific hurdle, the implementation of remote rendering has emerged as a viable solution. Hence, the primary objective of this thesis revolves around investigating and analyzing remote rendering methodologies within wireless virtual reality systems. To initiate this comprehensive study, the first stage entails an extensive collection of information encompassing diverse aspects such as methods, challenges, data transmission techniques, optimization strategies, and the evaluation of remote rendering. This meticulous examination aims to lay a solid foundation for understanding the intricacies associated with remote rendering. Furthermore, the study delves into the analysis of two distinct remote rendering systems employed in wireless VR. By scrutinizing these systems, the aim is to gain valuable insights into effectively addressing the limitations posed by wireless HMDs and devising applications that boast superior graphical quality. This analysis offers valuable guidelines and potential avenues for developing cutting-edge applications within the wireless virtual reality domain. In addition to the theoretical exploration, this thesis also encompasses the description of various tools that can be harnessed to achieve efficient remote rendering. These tools serve as indispensable resources in the implementation and optimization of remote rendering methodologies, providing researchers and developers with the means to enhance the overall VR experience. Finally, to showcase the practical implications of the aforementioned tools and methodologies, an application is developed. This practical implementation serves as a testament to the efficacy and potential of remote rendering in wireless virtual reality systems. Building upon the findings and observations derived from the application development, insightful conclusions are drawn, and thought-provoking suggestions are put forth to guide future research endeavors in this field.

Κεφάλαιο 1:Εισαγωγή

Η σημερινή κοινωνία παρουσιάζει ραγδαία τεχνολογική ανάπτυξη σε όλες τις πτυχές της, προσφέροντας πρόσφορο έδαφος σε καινούργιες τεχνολογίες να αναπτυχθούν. Μια από αυτές τις τεχνολογίες είναι η εικονική πραγματικότητα (Virtual reality), η οποία βρίσκει πολλές εφαρμογές σε διάφορους τομείς της καθημερινότητας όπως στην υγεία, στην ψυχαγωγία, στην εκπαίδευση κτλ. Εκπαιδεύονται διάφοροι επαγγελματίες και επενδύουν στην χρήση συστημάτων και μεθόδων εικονικής πραγματικότητας με σκοπό την βελτίωση των υπηρεσιών τους και κατ' επέκταση του βιοτικού επιπέδου. Τα υπάρχοντα συστήματα εικονικής πραγματικότητας μπορούν να χωριστούν σε δύο μεγάλες κατηγορίες, τα ενσύρματα και τα ασύρματα. Τα ενσύρματα συστήματα είναι ευρέως διαδεδομένα καθώς οι εφαρμογές τους χρησιμοποιούν υψηλής ποιότητας γραφικά λόγω της υψηλής υπολογιστικής ισχύς των υπολογιστών στους οποίους είναι συνδεδεμένα.

Βέβαια η χρήση των ενσύρματων συστημάτων εικονικής πραγματικότητας δεν είναι πανάκεια και παρουσιάζει ορισμένες αδυναμίες που καλούμαστε να αντιμετωπίσουμε. Μια βασική αδυναμία είναι η περιορισμένη δυνατότητα κίνησης που προσφέρουν στον χρήστη λόγω του καλωδίου, το οποίο μπορεί να οδηγήσει σε κάποιο ατύχημα όπως ο χρήστης να σκοντάψει ή να τυλιχτεί το καλώδιο γύρω από τον λαιμό του. Για την αντιμετώπιση αυτού του προβλήματος αναπτύχθηκαν ασύρματα συστήματα εικονικής πραγματικότητας χωρίς όμως να είναι και αυτά τέλεια. Το μεγαλύτερο πρόβλημα που αντιμετωπίζουν είναι το γεγονός ότι εφόσον είναι ανεξάρτητα του υπολογιστή και δεν συνδέονται με κάποιο καλώδιο δεν μπορούν να χρησιμοποιήσουν την υπολογιστική ισχύ του και χρησιμοποιούν την δικιά τους η οποία δεν επαρκεί κάτι που επηρεάζει την ποιότητα των γραφικών που προκύπτουν με σοβαρό αντίκτυπο στο τελικό αποτέλεσμα. Μια λύση για το παραπάνω πρόβλημα είναι η απομακρυσμένη απόδοση (remote rendering). Με άλλα λόγια η απόδοση πραγματοποιείται πρώτα σε έναν υπολογιστή και στην συνέχεια το αποτέλεσμα μεταδίδεται στην προαναφερθείσα ασύρματη συσκευή.

Στην παρούσα διπλωματική εργασία καλούμαστε να συγκεντρώσουμε πληροφορίες για την απομακρυσμένη απόδοση και να πειραματιστούμε αναπτύσσοντας μια εφαρμογή VR με σκοπό την διαχείριση υδατικών πόρων. Εδώ πρέπει να σημειωθεί ότι αυτή η εφαρμογή, παρόλο που στην συγκεκριμένη διπλωματική εργασία θα χρησιμοποιηθεί για το κομμάτι των υδατικών πόρων, μπορεί να είναι χρηστική και να βοηθήσει ένα πλήθος καθημερινών αλλά και εξειδικευμένων αναγκών.

Το πλεονέκτημα της εφαρμογής ότι έχει την δυνατότητα να μεταδώσει δύο επικαλυπτόμενες εικόνες (μία για κάθε μάτι) μίας τρισδιάστατης σκηνής από έναν server σε ένα ασύρματο HMD και να προκύπτει ως αποτέλεσμα μια στερεοσκοπική εικόνα την οποία ο χρήστης θα μπορεί να την δει μέσα από το HMD καθώς και να μεταδώσει δεδομένα εισόδου από το HMD στον server ώστε ο χρήστης να μπορεί να πραγματοποίηση κινήσεις στην τρισδιάστατη σκηνή. Το αποτέλεσμα της εφαρμογής ιδανικά θα πρέπει να υποστηρίζει ταχύτητα 90 frame per second για να αποφύγει το motion sickness.

Η προσέγγιση που θα πραγματοποιηθεί συνιστάται από τέσσερα βήματα τα οποία έχουν επιλεγεί με τέτοιο τρόπο ώστε να υπάρχει μια συνεχόμενη ροή και ο συνδυασμός τους να δύναται να δώσει τα απαραίτητα συμπεράσματα. Αρχικά, το πρώτο βήμα είναι η αναλυτική παρουσίαση του θεωρητικού υποβάθρου που χρησιμοποιήθηκε και πως αυτό συνδυάστηκε και χρησιμοποιήθηκε για τους σκοπούς της διπλωματικής εργασίας. Συγκεκριμένα θα αναφερθούμε στις game engines, στην απομακρυσμένη απόδοση, στις βασικές προκλήσεις της καθώς και σε τεχνικές βελτιστοποίησης της, αξιολογήσεις της και τρόπους μεταδώσεις. Ακόμα θα αναλυθούν δύο συστήματα remote rendering σε ασύρματα VR. Το δεύτερο βήμα αναφέρεται σε διάφορα εργαλεία που μπορούν να χρησιμοποιηθούν για να πετύχουμε τον σκοπό μας καθώς και σε αυτά που θα χρησιμοποιήσουμε. Το τρίτο βήμα συνιστάται από την τεχνική υλοποίηση της εφαρμογής, την λειτουργικότητα της και την αξιολόγηση της. Το τέταρτο και τελευταίο βήμα είναι τα συμπεράσματα που προκύπτουν αλλά και κατευθύνσεις για μελλοντικούς πειραματισμούς.

Κεφάλαιο 2: Βιβλιογραφική ανασκόπηση

Ορολογία Υπολογιστών	Επεξήγηση
Bandwidth	Το bandwidth ορίζει το πόσα δεδομένα μπορούν ενδεχομένως να μεταδοθούν σε μια δεδομένη χρονική στιγμή.
Client	Είναι κάποιο hardware υπολογιστή ή κάποιο λογισμικό το οποίο έχει πρόσβαση σε μια υπηρεσία που διατίθεται από έναν διακομιστή.
Cross-platform	Λογισμικό υπολογιστή το οποίο είναι σχεδιασμένο να λειτουργεί σε αρκετές διαφορετικές πλατφόρμες υπολογιστή.
Hardware	Είναι τα φυσικά κομμάτια ενός υπολογιστή.
Latency	Εκφράζει το χρόνο που χρειάζεται για να μεταδοθεί ένα πακέτο δεδομένων από ένα καθορισμένο σημείο σε ένα άλλο.
Server	Είναι κάποιο hardware υπολογιστή ή κάποιο λογισμικό που παρέχει λειτουργικότητα σε άλλα προγράμματα ή συσκευές.
Thin client	Υπολογιστές με χαμηλή υπολογιστική ισχύ που πρέπει να βασίζονται σε κάποιον server.
Thick client ή Fat client	Υπολογιστές που δεν χρειάζονται υπολογιστική δύναμη από κάποιον server.
Throughput	Το Throughput αναφέρεται στο πόσα δεδομένα μεταφέρονται πραγματικά κατά την διάρκεια μιας χρονικής περιόδου.
Vertical synchronization (VSync)	Είναι μια επιλογή στα περισσότερα συστήματα στα οποία η κάρτα γραφικών δεν μπορεί να κάνει οτιδήποτε ορατό στην οθόνη έως ότου η οθόνη ολοκληρώσει τον τρέχοντα κύκλο ανανέωσης.
Rendering	Το rendering είναι η διαδικασία παραγωγής μιας εικόνας από ένα μοντέλο 2D ή 3D μέσω ενός προγράμματος υπολογιστή.

Πίνακας 1: Ορισμοί ορολογίας υπολογιστών

2.1 Remote rendering

Ένα σύστημα απομακρυσμένης απόδοσης (remote rendering) εκτελεί εφαρμογές απόδοσης σε μία υπολογιστική συσκευή, που αναφέρεται ως διακομιστής (server) ή διακομιστής απόδοσης (rendering server) και εμφανίζει τα αποτελέσματα απόδοσης (rendering) σε μια άλλη υπολογιστική συσκευή συνδεδεμένη στο δίκτυο, η οποία αναφέρεται ως client. Ένα διαδραστικό σύστημα remote rendering είναι ικανό να

δέχεται στοιχεία ελέγχου χρήστη μέσω των συσκευών εισόδου σε έναν client για αλληλεπίδραση με τις εφαρμογές rendering που εκτελούνται σε server.

Η έννοια του remote rendering εμφανίστηκε από το 1999 [Ohazama 1999] όταν οι υπολογιστές δεν ήταν αρκετά ισχυροί για την πραγματοποίηση του rendering τρισδιάστατων γραφικών. Τα τελευταία χρόνια, η ευρεία ανάπτυξη ασύρματων δικτύων υψηλής ταχύτητας και οι εξελίξεις σε τεχνολογίες φορητού υπολογιστικού νέφους (mobile cloud computing) έχουν εντείνει την έρευνα για remote rendering. Οι Beermann και Humphreys [2003] προέβλεψαν ότι το rendering γραφικών θα γινόταν μια απομακρυσμένη υπηρεσία και αυτή η πρόβλεψη επαληθεύτηκε με την εμφάνιση του cloud gaming.

Συγκριτικά με τη συμβατική προσέγγιση που εκτελεί την απόδοση τοπικά (local rendering), το remote rendering έχει πολλά πλεονεκτήματα. Πρώτον, μπορεί να προσφέρει πλούσιες εμπειρίες απόδοσης σε "thin" clients (π.χ. κινητές συσκευές) με περιορισμένους υπολογιστικούς πόρους (π.χ. GPU, μνήμη, μπαταρία). Δεύτερον, οι υπολογιστικοί πόροι σε έναν rendering server μπορούν να μοιράζονται αποτελεσματικά από πολλούς clients. Τρίτον, το remote rendering είναι απλή αλλά αποτελεσματική λύση cross-platform. Είναι χαρακτηριστικό ότι, μετά την ανάπτυξη των προγραμμάτων client για διαφορετικές πλατφόρμες, όλες οι εφαρμογές πρέπει να αναπτυχθούν μόνο για τον server και οποιαδήποτε ενέργεια rendering θα επιτευχθεί σε όλες τις πλατφόρμες των client. Τέλος, ένα σύστημα remote rendering μπορεί να σχεδιαστεί για να αποτρέπει τη διαρροή (leaking) του περιεχομένου της πηγής σε κακόβουλους χρήστες με ροή (streaming), αποδίδοντας αποτελέσματα rendering μόνο στους επιθυμητούς clients. Για παράδειγμα στο cloud gaming, οι επί του παρόντος αναπτυσσόμενες λύσεις remote rendering μπορούν να αποτρέψουν ανησυχίες για την πειρατεία επειδή το περιεχόμενο του παιχνιδιού δεν αποστέλλεται ποτέ σε παίκτες.

2.1.1 Βασικές προκλήσεις

Διαφορετικά συστήματα remote rendering έχουν σχεδιαστεί για να στοχεύουν σε διάφορα προβλήματα και προκλήσεις. Στην συνέχεια θα επικεντρωθούμε σε δύο βασικά προβλήματα: πώς να μειώσουμε την καθυστέρηση αλληλεπίδρασης (interaction latency) και πώς να μεταδώσουμε δεδομένα από server σε client αποτελεσματικά και αποδοτικά.

Αρχικά ορίζουμε το "interaction latency" ως το χρόνο από τη δημιουργία ενός αιτήματος αλληλεπίδρασης από τον χρήστη μέχρι την εμφάνιση του πρώτου ενημερωμένου πλαισίου (frame) στην οθόνη του client. Ως ένα εγγενές πρόβλημα όλων των κατανεμημένων συστημάτων δικτύου, η καθυστέρηση (latency) παίζει σημαντικό ρόλο στην ποιότητα της υπηρεσίας QoS (Quality of Service) και στην ποιότητα της εμπειρίας QoE (Quality of Experience). Η μεγάλη καθυστέρηση αλληλεπίδρασης (interaction latency) μπορεί να υποβαθμίσει σημαντικά την εμπειρία του χρήστη. Για παράδειγμα, τα 100 ms είναι η μεγαλύτερη ανεκτή καθυστέρηση (latency) για ένα εντατικό παιχνίδι σκοποβολής πρώτου προσώπου. Επομένως, η ελαχιστοποίηση αυτής είναι καίρια ανησυχία και θα εξετάσουμε όλες τις σχετικές τεχνικές μείωσης του latency που μπορούν να εφαρμοστούν στο remote rendering.

Η μετάδοση δεδομένων είναι πάντα το βασικό δομοστοιχείο (module) οποιουδήποτε συστήματος remote rendering. Στις περισσότερες περιπτώσεις, ένας σχεδιασμός module μετάδοσης δεδομένων θα πρέπει να λαμβάνει υπόψη όλα τα είδη περιορισμών, όπως περιορισμένο bandwidth, διάφορες συνθήκες καναλιών και απαιτήσεων σε πραγματικό χρόνο. Αναμφίβολα η επιτυχία της ροής δεδομένων (data streaming) έχει άμεσο αντίκτυπο στη συνολική απόδοση του συστήματος.

2.1.2 Παραπλήσιες τεχνικές

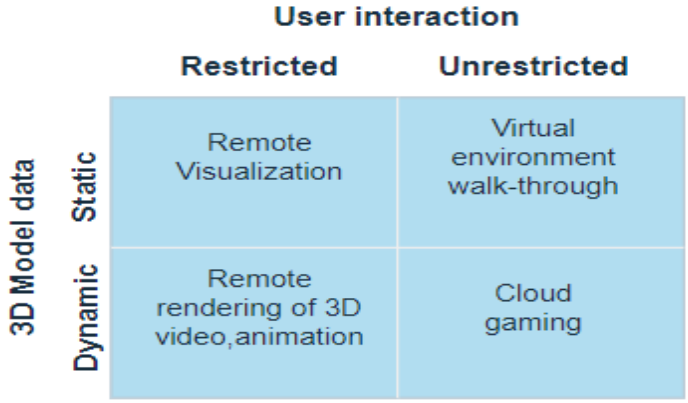
Thin Clients και Remote Sharing. Τα συστήματα Thin Client (π.χ. SLIM [Schmidt et al. 1999], THiNC [Baratto et al. 2005], κ.λπ.) και τα συστήματα κοινής χρήσης απομακρυσμένης επιφάνειας εργασίας (π.χ. VNC [Richardson et al. 1998], RDP [Cumberland et al. 1999], κ.λπ.) επιτρέπουν στους χρήστες να έχουν πρόσβαση σε εφαρμογές εξ αποστάσεως και να έχουν κοινή χρήση υπολογιστικών πόρων. Οι χρήστες από την πλευρά του client μπορούν να αλληλοεπιδράσουν με τις εφαρμογές που εκτελούνται από την πλευρά του server. Ωστόσο, υπάρχουν βασικές διαφορές μεταξύ αυτών των συστημάτων και των συστημάτων remote rendering που θα εξετάσουμε.

Πρώτον, τα περισσότερα συστήματα thin client και remote sharing εμφανίστηκαν πριν γίνει δημοφιλές το rendering 3D γραφικών. Τα πρώτα συστήματα είχαν σχεδιαστεί μόνο για να υποστηρίζουν κοινή χρήση επιτραπέζιων υπολογιστών και rendering 2D γραφικών. Μόνο οι πρόσφατες προσεγγίσεις άρχισαν να προσθέτουν υποστήριξη για τρισδιάστατα γραφικά (π.χ. THiNC [Baratto et al. 2005], TurboVNC [Commander 2007], και τα λοιπά.). Δεύτερον, ο κύριος ερευνητικός στόχος της κοινής χρήσης 2D εφαρμογών είναι ο σχεδιασμός πρωτοκόλλων που μπορούν να ενημερώσουν αποτελεσματικά τις περιφερειακές αλλαγές στην οθόνη. Αυτό συμβαίνει επειδή η απόδοση 2D θεωρείται ελαφριά, έτσι ώστε να μπορεί να πραγματοποιηθεί είτε σε server είτε σε client. Ωστόσο, αυτή η υπόθεση δεν ισχύει για τα τρισδιάστατα γραφικά λόγω της σημαντικά αυξημένης πολυπλοκότητας του rendering. Τέλος, οι εφαρμογές 3D όπως τα βιντεοπαιχνίδια συνήθως ανανεώνουν ολόκληρη την οθόνη με πολύ υψηλότερο ρυθμό, συνήθως μεγαλύτερο από 60 fps. Εφαρμογές σαν αυτές απαιτούν επίσης διαφορετικές μεθόδους συμπίεσης (compression) και streaming που είναι διαφορετικές από αυτές που χρησιμοποιούνται στο συμβατικό 2D συστήματα thin client.

2.1.3 Βασική Κατηγοριοποίηση

Σύμφωνα με τον Shi [2015] κατηγοριοποιείται το remote rendering με βάση ένα πίνακα δύο διαστάσεων ο οποίος περιέχει ως γραμμές τον τύπο δεδομένων του τρισδιάστατου μοντέλου και ως στήλες το τύπο της αλληλεπίδρασης του χρήστη (εικόνα 1). Τα δεδομένα του τρισδιάστατου μοντέλου (3D model data) είναι είτε δυναμικά (dynamic) είτε στατικά (static), ανάλογα με το αν ενημερώνονται συνεχώς κατά τη διάρκεια των εκτελέσεων. Οι αλληλεπιδράσεις των χρηστών (user interaction)

είναι είτε περιορισμένες (restricted) είτε χωρίς περιορισμούς (unrestricted), ανάλογα με το αν οι χρήστες είναι εντελώς ελεύθεροι να αλλάξουν τη θέση της κάμερας, τον προσανατολισμό και άλλες ενέργειες. Οι κατηγορίες remote rendering σε διαφορετικές περιπτώσεις επιβάλλουν διαφορετικούς σχεδιασμούς.



Εικόνα 1: Κατηγορίες απομακρυσμένης απόδοσης

(Πηγή:https://www.researchgate.net/publication/277967155_A_Survey_of_Interactive_Remote_Rendering_Systems Shi et al [2015])

Στατικά μοντέλα και περιορισμένη αλληλεπίδραση.

Τα συστήματα που ανήκουν σε αυτή την κατηγορία αποδίδουν στατικά τρισδιάστατα μοντέλα και υποστηρίζουν περιορισμένες αλληλεπιδράσεις από τους χρήστες. Ο όρος "περιορισμένη αλληλεπίδραση χρήστη" σημαίνει ότι ένας χρήστης μπορεί να αλληλοεπιδράσει με τα τρισδιάστατα δεδομένα μόνο με μερικούς τρόπους. Για παράδειγμα, ένας χρήστης μπορεί να δει ένα τρισδιάστατο μοντέλο μόνο από μερικές προκαθορισμένες οπτικές γωνίες ή να αλλάξει την οπτική γωνία σε ένα περιορισμένο εύρος. Τα περισσότερα συστήματα απομακρυσμένης οπτικοποίησης (remote visualization) ανήκουν σε αυτή την κατηγορία. Τα συστήματα αυτού του τύπου χρησιμοποιούνται ευρέως για την παρουσίαση διαφόρων τύπων δεδομένων, όπως ιατρικές εικόνες, επιστημονικά δεδομένα, σχέδια βιομηχανίας και έργα τέχνης. Πολλά από τα συστήματα visualization εκτελούν την απόδοση εξ αποστάσεως λόγω της πολυπλοκότητας της πηγής των δεδομένων ή τους περιορισμούς των συσκευών παρουσίασης.

Το remote visualization έχει επίσης εφαρμοστεί για να παρέχει πλούσιες εμπειρίες rendering σε λιγότερο ισχυρές συσκευές. Η Diepstraten et al. [2004] πρότεινε τη δημιουργία ενός απλοποιημένου μοντέλου το οποίο δημιουργεί γραφικά μόνο με την χρήση γραμμών για clients που δεν διαθέτουν υλικό (hardware) τρισδιάστατων γραφικών, αλλά μπορούν να εκτελέσουν λειτουργιές κανονικοποίησης (rasterization) 2D. Οι Duguet και Drettakis [2004] μετέτρεψαν τα τρισδιάστατα μοντέλα σε νέφη σημείων χαμηλότερης ανάλυσης για clients με μικρές οθόνες. Ο Levoy [1995] πρότεινε μια υβριδική λύση για τη δημιουργία μιας απλοποιημένης αναπαράστασης πολυγώνων χαμηλής ποιότητας ως εικόνα διαφοράς (difference image), αποφεύγοντας την απώλεια ποιότητας rendering από την πλευρά του client. Mann και Ο Cohen-Or [1997]

σχεδίασαν ένα παρόμοιο σύστημα που μεταδίδει τόσα πολύγωνα όσο και εικονοστοιχεία (pixel), αλλά επικεντρώθηκαν στην επιλογή των πιο σημαντικών pixel για εξοικονόμηση bandwidth.

Το remote visualization που βασίζεται στο Web απευθύνεται σε έναν ειδικό τύπο «thin» client. Ειδικότερα δίνει τη δυνατότητα σε οποιοσδήποτε client να έχει πρόσβαση στις υπηρεσίες rendering μέσω ενός προγράμματος περιήγησης στο διαδίκτυο (web browser). Οι περιορισμοί τέτοιων συστημάτων περιλαμβάνουν το γενικό κόστος των τυπικών επικοινωνιών ιστού (overhead of web communication) και την άγνωστη υπολογιστική ικανότητα που είναι διαθέσιμη στον client. Το Reality server [Nvidia 2009] από τη Nvidia κυκλοφόρησε για να παρέχει υπηρεσίες φωτορεαλιστικού rendering μέσω του web. Οι Yoon και Neumann [2000] σχεδίασαν ένα σύστημα βασισμένο σε web χρησιμοποιώντας τεχνικές IBR (Image-Based Rendering).

Το remote visualization που βασίζεται σε κινητά (mobile) απευθύνεται σε έναν άλλο τύπο «thin» client: κινητές συσκευές (π.χ. smartphone και tablet). [Woodward et al. 2002] και [D'Amora και Bernardini 2003] ήταν δύο πρώιμες απόπειρες χρήσης ενός PDA ως CAD (Computer-Aided Design) προβολής ενώ η πραγματική εφαρμογή CAD εκτελούνταν σε έναν απομακρυσμένο διακομιστή (remote server). Ο Chang και ο Ger [2002] πρότειναν την κατασκευή LDIs (Layered Depth Images) [Shade et al. 1998] στον server. Ο mobile client εκτελεί αλγόριθμους IBR για να συνθέσει την εικόνα εμφάνισης (display image). Τέτοιες μελέτες μπορεί να φαίνονται "ξεπερασμένες" τώρα, επειδή τα τσιπ για κινητά έχουν βελτιωθεί δραστικά την τελευταία δεκαετία. Οι τρέχουσες κινητές συσκευές είναι εξοπλισμένες με σχετικά ισχυρά τσιπ GPU πολλαπλών πυρήνων για rendering σύνθετων 3D γραφικών. Ωστόσο, οι περιορισμοί στις κινητές συσκευές εξακολουθούν να υφίστανται για εφαρμογές υψηλής υπολογιστικής έντασης, όπως βιντεοπαιχνίδια.

Τα συστήματα remote visualization μπορούν να εξυπηρετήσουν άλλους σκοπούς. Οι Koller et al. [2004] ανέπτυξαν ένα σύστημα remote rendering για να αποτρέψει τα πολύτιμα δεδομένα τρισδιάστατων μοντέλων που σαρώθηκαν από διάσημα έργα τέχνης από τη διαρροή σε κακόβουλους χρήστες. Αυτό το σύστημα ακολούθησε μια προσέγγιση ροής εικόνας (image streaming) στέλνοντας μόνο τα αποτελέσματα rendering στον χρήστη, προστατεύοντας αποτελεσματικά την πηγή των τρισδιάστατων δεδομένων.

Δυναμικά μοντέλα και περιορισμένη αλληλεπίδραση.

Αξίζει να αναφερθεί πως, για ορισμένες εφαρμογές όπως το τρισδιάστατο βίντεο ή το animation, τα τρισδιάστατα δεδομένα δημιουργούνται ή ενημερώνονται δυναμικά και η σκηνή συνεχώς αλλάζει. Το rendering τέτοιων εφαρμογών απαιτεί την επεξεργασία κάθε frame σε πραγματικό χρόνο πριν από την άφιξη του επόμενου frame. Αυτό σημαίνει πως, καμία προετοιμασία εκτός σύνδεσης ή χρονοβόρος υπολογισμός στον server (π.χ. εκτέλεση χρονοβόρων αλγορίθμων απλοποίησης πολυγώνων ή δημιουργία LDI για το μοντέλο) μπορεί να εφαρμοστεί σε αυτήν την κατηγορία συστημάτων remote rendering.

Επιπρόσθετα, Shi et al. [2008] πρότεινε απόδοση εικόνων βάθους για τρισδιάστατο τηλε-εμβυθιστικό βίντεο (tele-immersive video) και αξιοποίησε αλγόριθμους IBR για να μειώσει την latency αλλαγής οπτικής γωνίας [Shi et al. 2009; Shi et al. 2010]. Ο

Lamberti et al το [2003] παρουσίασε ένα remote rendering framework που βασίζεται σε εικόνες για mobile client .Επιπλέον, ο Humphreys et al. [2002] ανέπτυξε ένα βελτιωμένο σύστημα video streaming που χρησιμοποιούσε συμπλέγματα απόδοσης (rendering clusters) όχι μόνο για τρισδιάστατο rendering αλλά επίσης για κωδικοποίηση βίντεο σε πραγματικό χρόνο.

Στατικά μοντέλα και μη περιορισμένη αλληλεπίδραση.

"Μη περιορισμένη αλληλεπίδραση" σημαίνει ότι ο χρήστης μπορεί να αλλάξει ελεύθερα την οπτική γωνία στον τρισδιάστατο χώρο και μπορεί να πραγματοποιήσει επιπλέον ενέργειες που μπορούν να αλλάξουν τα τρισδιάστατα δεδομένα. Η αντιπροσωπευτική εφαρμογή αυτής της κατηγορίας είναι η πλοήγηση σε εικονικό περιβάλλον VE (Virtual Environment). Για την ακρίβεια, τα συστήματα πλοήγησης VE συνήθως χρησιμοποιούνται σε διαδικτυακά παιχνίδια και συστήματα εικονικής πραγματικότητας VR (Virtual Reality) [Singhal and Zyda 1999]. Ωστόσο, η κύρια πρόκληση της πλοήγησης VE είναι ότι το γραφικό μοντέλο ολόκληρου του VE είναι πολύ μεγάλο για να μεταδοθεί αρκετά γρήγορα για διαδραστική απόδοση (interactive rendering) σε πραγματικό χρόνο. Κατόπιν, υπάρχουν δύο κύριες κατευθύνσεις VE. Η μια κατεύθυνση εστιάζει σε peer-to-peer επικοινωνίες μεταξύ πολλών χρηστών για τη συγχρονισμένη απόδοση του εικονικού κόσμου [Funkhouser 1995] και η άλλη μελετά πώς να μεταδώσει (stream) αποτελεσματικά τον εικονικό κόσμο από τον κεντρικό server στους τελικούς χρήστες [Cohen-Or et al. 2003].

Ο Schmalstieg [1997] πρότεινε τη χρήση ενός remote rendering pipeline για τη διαχείριση της γεωμετρίας και του bandwidth σε ένα κατανεμημένο VE. Πιο συγκεκριμένα, το pipeline χρησιμοποιεί τον server ως βάση δεδομένων για τα γραφικά μοντέλα και μεταδίδει μόνο τα μοντέλα που είναι ορατά από την οπτική γωνία του χρήστη. Επιπλέον, η βάση δεδομένων διακομιστή (server database) οργανώνει όλα τα τρισδιάστατα μοντέλα γραφικών ως διαφορετικά επίπεδα λεπτομέρειας LOD (levels of detail) και τα αποθηκεύει σε δομές Oct-Tree. Για διαφορετικές περιοχές ενδιαφέροντος AOI (areas of interest), μεταδίδονται διαφορετικά LOD.

Επίσης, υπάρχουν συστήματα VE που ακολουθούν την προσέγγιση image streamming. Το QuickTime σύστημα VR [Chen 1995] δημιουργεί μια πανοραμική εικόνα 360 μοιρών του κόσμου με βάση την τοποθεσία του χρήστη. Αυτός ο χάρτης περιβάλλοντος επιτρέπει στον client να περιηγηθεί χωρίς κανένα ειδικό υλικό γραφικών (graphical hardware) ή περαιτέρω ενημερώσεις από τον server. Οι Boukerche και Pazzi [2006] ακολούθησαν την ίδια κατεύθυνση και ανέπτυξαν ένα κινητό σύστημα πλοήγησης VE στο πανοραμικό rendering. Οι Bao και Gourlay [2004] σχεδίασαν ένα σύστημα περιήγησης VE χρησιμοποιώντας μια διαφορετική προσέγγιση που βασίζεται σε εικόνες. Στο σύστημά τους, ο server αποδίδει την εικονική σκηνή και στέλνει την εικόνα του αποτελέσματος με τον χάρτη βάθους στον client. Ο client μπορεί να εμφανίσει τις εικόνες ή να εκτελέσει IBR αλγόριθμους για να συνθέσει νέες εικόνες όταν αλλάζει η οπτική γωνία που θα γίνει render. Οι Noimark και Cohen-Or [2003] εισήγαγαν ένα MPEG-4 Σύστημα περιήγησης VE βασισμένο σε streaming. Οι σκηνές VE γίνονται render συνεχώς στον server με προκαθορισμένο ρυθμό ανανέωσης καρέ (frame rate) και κωδικοποιούνται σε πραγματικό χρόνο χρησιμοποιώντας το πρότυπο MPEG-4.Αυτή η προσέγγιση δεν χρειάζεται να αναλύσει τα τρισδιάστατα

μοντέλα πηγής για τον προσδιορισμό του LOD, αλλά απαιτεί σταθερό bandwidth δικτύου μεταξύ server και client για video streaming. Οι Lamberti et al. (2003) και οι Quax et al. (2006) και οι δύο πρότειναν παρόμοια απόδοση βασισμένη σε βίντεο για χρήστες κινητών.

Δυναμικά μοντέλα και μη περιορισμένη αλληλεπίδραση.

Το Cloud gaming είναι η αντιπροσωπευτική εφαρμογή αυτής της τελευταίας κατηγορίας. Η εμφάνιση των υπηρεσιών cloud gaming έχει μετακινήσει τους υπολογισμούς rendering τρισδιάστατων βιντεοπαιχνιδιών στο cloud. Σε σύγκριση με άλλες προσεγγίσεις το remote rendering βιντεοπαιχνιδιών είναι ένα πολύ πιο δύσκολο έργο. Συγκεκριμένα , τα πιο πρόσφατα βιντεοπαιχνίδια απαιτούν πολύ περίπλοκο τρισδιάστατο rendering γραφικών για την παρουσίαση του εικονικού κόσμου. Επομένως, το Cloud gaming απαιτεί προηγμένο υλικό γραφικών (graphical hardware) για rendering. Επίσης, τα παιχνίδια πρέπει να γίνονται rendered με υψηλό frame rate, υψηλότερο από 60 frames το δευτερόλεπτο fps (frame per second) για να αντιμετωπίσουν τις δυναμικά μεταβαλλόμενες σκηνές και τις έντονες κινήσεις. Ιδιαίτερα σημαντικό αναφοράς είναι πως, το υψηλό latency είναι απαγορευτικό. Αρκετές εταιρείες έχουν παράξει υπηρεσίες και λύσεις cloud gaming, όπως το OnLive, το οποίο χρησιμοποιεί μια προσέγγιση video streaming που κάνει render βιντεοπαιχνίδια στο cloud και στέλνει σκηνές παιχνιδιού (gameplay) ως video stream ανάλυσης 720p στους τελικούς χρήστες.

Καταρχήν, Game@Large [Eisert and Fechteler 2008; Nave et al. 2008] πρότειναν την αποστολή δεδομένων τρισδιάστατων γραφικών στον client. Είναι γεγονός ότι λειτουργίες rendering γραφικών και οι υφές (texture) συμπιέζονται αποτελεσματικά και μεταδίδονται χρησιμοποιώντας το πρωτόκολλο RTP (Real-Time Transport Protocol). Κατόπιν, ο Jurgelionis et al. [2009] βελτίωσε το Game@Large με μια υβριδική προσέγγιση. Για "fat" clients, όλες οι λειτουργίες rendering γραφικών και τα textures γίνονται stream όπως το αρχικό σχέδιο του Game@Large. Για "thin" clients που δεν έχουν αρκετή υπολογιστική ισχύ για rendering τρισδιάστατων γραφικών, το rendering παιχνιδιών πραγματοποιείται στον server και οι τελικές εικόνες συμπιέζονται με H.264 [Wiegand et al. 2003] για τον client. Το GamingAnywhere [Huang et al. 2013] είναι μια ανοιχτή πλατφόρμα cloud gaming, η οποία μπορεί να χρησιμοποιηθεί από ερευνητές, προγραμματιστές και παίκτες για δοκιμές. Είναι δυνατές διάφορες προσαρμογές στο GamingAnywhere, όπως η απόθεση ενός κωδικοποιητή (codec) H.264/MVC [Vetro et al. 2011] για την υποστήριξη στερεοσκοπικών παιχνιδιών.

Κατηγορίες	Μέθοδοι
Στατικά μοντέλα και περιορισμένη αλληλεπίδραση	Δημιουργία γραφικών με χρήση γραμμών. Μετατροπή τρισδιάστατων μοντέλων σε νέφη σημείων. Αναπαράσταση πολυγώνων χαμηλής ποιότητας ως εικόνα διαφοράς. Φωτορεαλιστικό rendering μέσω του web (Reality server) Συστήματα βασισμένα σε web χρησιμοποιώντας τεχνικές IBR. Image Streaming.
Δυναμικά μοντέλα και περιορισμένη αλληλεπίδραση	Απόδοση εικόνων βάθους και χρήση αλγόριθμων IBR για μείωση του latency στην αλλαγή της οπτικής γωνίας. Video streaming χρησιμοποιώντας συμπλέγματα απόδοσης.
Στατικά μοντέλα και μη περιορισμένη αλληλεπίδραση	Μετάδοση μοντέλων που είναι ορατά από την οπτική γωνία του χρήστη. Απόδοση εικονικής σκηνής και αποστολής εικόνας με τον χάρτη βάθους. Image Streaming.
Δυναμικά μοντέλα και μη περιορισμένη αλληλεπίδραση	Υπολογισμοί του rendering στο cloud.

Πίνακας 2: Μέθοδοι με βάση της κατηγορίες απομακρυσμένης απόδοσης

2.1.4 Τεχνικές βελτιστοποίησης Latency

Με τις συμβατικές τεχνικές βελτιστοποίησης (optimization), το συνολικό interaction latency δεν μπορεί να περάσει το κατώτατο όριο καθυστέρησης ενός network round trip time – RTT (ο χρόνος που απαιτείται από τον server να φθάσει στον client και να επιστρέψει). Ωστόσο, είναι δυνατή η δημιουργία των οπτικών frame απόκρισης στον client πριν από οποιονδήποτε server.

Τοπική απόδοση και μηδενικός απολογισμός (Local Rendering and Dead Reckoning)

Αναμφίβολα, η ευκολότερη μέθοδος για τη δημιουργία ενός frame απόκρισης τοπικά είναι η αποθήκευση τρισδιάστατων δεδομένων στον client και η τοπική εκτέλεση του rendering 3D γραφικών. Μερικοί περιηγητές VE [Lluch et al. 2005] και συστήματα remote visualization [Duguet και Drettakis 2004] που μεταδίδουν meshes στον client μπορούν να εφαρμόσουν το local rendering ώστε να μειώσουν το interaction latency στο χρόνο που απαιτείται για το rendering τρισδιάστατων γραφικών σε κάθε συσκευή των client. Επιπρόσθετα, ο πάροχος παιχνιδιών cloud Ciinow ισχυρίζεται επίσης ότι η ενσωμάτωση της ροής τρισδιάστατων γραφικών [Dharmapurikar 2013b] μειώνει το latency.

Ένα βασικό ζήτημα σε αυτήν την προσέγγιση είναι να διατηρείται το local rendering συγχρονισμένο με τον server. Για εφαρμογές visualization ή περιήγησης, ο συγχρονισμός server-client μπορεί να χρειάζεται μόνο να διατηρεί σταθερή την οπτική γωνία που γίνεται render. Συνεπώς ο συγχρονισμός γίνεται πιο περίπλοκος για εφαρμογές παιχνιδιών επειδή το rendering σκηνών παιχνιδιών απαιτεί γνώση της λογικής του παιχνιδιού και των δεδομένων χρήστη (π.χ. την ταχύτητα και την κατεύθυνση των κινούμενων αντικειμένων).

Dead Reckoning, μια στρατηγική που υιοθετείται ευρέως για την ανάπτυξη διαδικτυακών παιχνιδιών για πολλούς παίκτες [Pantel and Wolf 2002], μπορεί να εφαρμοστεί για να μετριάσει το πρόβλημα συγχρονισμού. Πιο συγκεκριμένα η βασική έννοια του dead reckoning είναι ένα σύνολο αλγορίθμων που μπορούν να χρησιμοποιηθούν από τον client για την προεκβολή της συμπεριφοράς οντοτήτων στο παιχνίδι. Επίσης πρέπει να προσδιοριστεί το πόσο μακριά να επιτρέπεται να φτάσει η πραγματικότητα από αυτούς τους αλγόριθμους προεκβολής πριν εκδοθεί μια διόρθωση. Στο σενάριο του remote rendering μπορεί να γίνει διόρθωση με τις ενημερώσεις του server μετά από την καθυστέρηση ενός network round trip.

Προ-ανάκτηση (Pre-fetch)

Για ένα σύστημα που δεν μπορεί να επεξεργαστεί το rendering τρισδιάστατων γραφικών στην πλευρά του client, το pre-fetch είναι μια αποτελεσματική προσέγγιση μείωσης της latency. Ειδικότερα ένας client ή server προβλέπει τις μελλοντικές κινήσεις ενός χρήστη και ζητά από τον server να μεταδώσει τις εικόνες που αποδίδονται για όλες τις πιθανές κινήσεις. Αυτό σημαίνει ότι εάν η πρόβλεψη πετύχει, ο client μπορεί απλώς να εμφανίσει τις προ ανακτημένες εικόνες και δεν παρατηρείται latency. Εάν η πρόβλεψη αποτύχει, ο client θα πρέπει να αδειάσει την ενδιάμεση μνήμη προ-ανάκτησης (pre-fetch buffer) και περιμένει να φτάσουν οι σωστές εικόνες από τον server.

Οι Hesina και Schmalstieg [1998] εισήγαγαν ένα pre-fetch framework για ένα διαδυκτιακό VE. Ο Chen et al. [2008] συζήτησε τον τρόπο pre-fetch εικόνων πολλαπλών διαφορετικών αναλύσεων ώστε να καλύπτουν διαφορετικά επίπεδα λεπτομέρειας. Ο Touch [1995] εισήγαγε μια προσέγγιση προ-αποστολής για γενικές επικοινωνίες μέσω δικτύων υψηλής ταχύτητας. Η προ-αποστολή βοηθά στη μείωση της latency κατά το ήμισυ του χρόνου καθυστέρησης network round trip όταν το bandwidth δεν δημιουργεί bottleneck.

Επιπρόσθετα, η pre-fetch λειτουργεί καλύτερα για την πλοήγηση σε ένα στατικό περιβάλλον καθώς η εικόνα μπορεί να αποθηκευτεί για μελλοντική χρήση. Η επίδοση (performance) των τεχνικών pre-fetch βασίζεται εξ ολοκλήρου στον ποσοστό επιτυχίας της πρόβλεψης της κίνησης και στο μέγεθος του pre-fetch buffer. Το κόστος είναι το επιπλέον bandwidth δικτύου που απαιτείται για τη μετάδοση όλων των pre-fetched εικόνων σκηνής όπου οι περισσότερες ενδέχεται να μην χρησιμοποιηθούν ποτέ. Η pre-fetch είναι ακατάλληλη για εφαρμογές που αποδίδουν δυναμικά δεδομένα, όπως παιχνίδια. Τέτοιες εφαρμογές ανανεώνονται γρήγορα και κάθε frame λήγει με την άφιξη του επόμενου frame. Επομένως, ο client πρέπει να εκτελέσει pre-fetch για κάθε νέο frame γεγονός που αυξάνει σημαντικά τη χρήση bandwidth δικτύου.

Απόδοση με βάση την εικόνα (Image-Based Rendering)

Το image-based rendering εμπίπτει μεταξύ της τρισδιάστατης προσέγγισης (local rendering) και της εικόνας προσέγγιση (pre-fetch). Απαιτεί από τον server να στείλει επιπλέον πληροφορίες με τις εικόνες των αποτελεσμάτων απόδοσης στον client. Στην πλευρά του client, όταν η αλληλεπίδραση του χρήστη αλλάζει την οπτική γωνία του rendering, οι επιπλέον πληροφορίες μπορούν να χρησιμοποιηθούν για την εκτέλεση αλγορίθμων IBR οι οποίοι συνθέτουν την display image στη νέα οπτική γωνία. Επομένως, η interaction latency μειώνεται στο χρόνο που απαιτείται για την εκτέλεση αλγορίθμων IBR.

Ένας πολύ γνωστός αλγόριθμος IBR είναι η στρέβλωση της τρισδιάστατης εικόνας (3D image warping) [McMillan 1997]. Παίρνει μια εικόνα βάθους (depth image), την οπτική γωνία που αντιστοιχεί στην depth image και την στοχευμένη οπτική γωνία (target viewpoint) ως δεδομένα εισαγωγής και εξάγει την εικόνα του target viewpoint Αυτός ο αλγόριθμος μπορεί να εκτελείται αποτελεσματικά σε ενσωματωμένες CPU/GPU [Yoo et al. 2010] ή σε προσαρμοσμένο υλικό [Popescu et al. 2000]. Το 3D image warping έχει εφαρμοστεί σε πολλά συστήματα remote rendering [Chang and Ger 2002; Shi et al. 2012a; Mark 1999; Bao και Gourlay 2004]. Για κάθε frame, ο server πρέπει να στείλει έναν χάρτη βάθους με μια εικόνα αποτελέσματος στον client και η interaction latency για οποιαδήποτε αλλαγή οπτικής γωνίας μπορεί να μειωθεί στον χρόνο που απαιτείται για την εκτέλεση του αλγόριθμου 3D image wrapping στη συσκευή του client.

Ωστόσο, η χρήση του αλγόριθμου 3D image warping δημιουργεί το πρόβλημα της "έκθεσης". Η εικόνα του αποτελέσματος της σύνθεσης περιέχει συνήθως κενά σημεία επειδή δεν υπάρχουν επαρκή pixel στην εικόνα εισόδου για να γεμίσει την επιφάνεια σε μια νέα οπτική γωνία. Έχουν προταθεί διάφορες τεχνικές για να γεμίσουν τα κενά σημεία. Τα φίλτρα βάθους [Redert et al. 2002] και splat warping [Mark 1999] είναι αποτελεσματικά για μικρές τρύπες. Super view warping [Bao and Gourlay 2004] και πλατύ field of view warping [Mark 1999] μπορεί να λύσει εν μέρει το πρόβλημα. LDI [Shade et al. 1998] και Double Warping [Shi et al. 2009; Shi et al. 2010] βασίζονται και τα δύο στην ιδέα του χρησιμοποιώντας πολλαπλές αναφορές για την κάλυψη εκτεθειμένων οπών.

Από όλες τις τεχνικές που αναφέρονται παραπάνω, το Double Warping είναι ίσως η πιο κατάλληλη τεχνική για τη μείωση της interaction latency σε συστήματα remote rendering. Με αυτήν την προσέγγιση, ένας server αποδίδει όχι μόνο την depth image στην τρέχουσα οπτική γωνία, αλλά πολλαπλές depth images σε άλλες οπτικές γωνίες. Ο client εκτελεί τον αλγόριθμο 3D image warping για όλες depth images και συνθέτει όλα τα αποτελέσματα. Οι οπτικές γωνίες επιλέγονται προσεκτικά με βάση μια πρόβλεψη της κίνησης της κάμερας [Mark 1999; Shi et al. 2010]. Η πρόβλεψη εδώ είναι ελαφρώς διαφορετική από την πρόβλεψη κίνησης του pre-fetch, δεν χρειάζεται να δώσει την ακριβή θέση της μελλοντικής οπτικής γωνίας, αλλά μόνο μια πιθανή κατεύθυνση/περιοχή. Με την προβλεπόμενη κατεύθυνση/περιοχή κίνησης, οι αναφορές θα πρέπει να επιλέγονται για να καλύπτουν αυτήν την περιοχή έτσι ώστε μια εικόνα υψηλής ανάλυσης χωρίς κενά σημεία να συντεθεί από τον client. Διαφορετικοί αλγόριθμοι επιλογής αναφοράς έχουν μελετηθεί στο [Mark 1999; Shi et al. 2009; Shi et al. 2012a; Hudson και Mark 1999; Οι Thomas et al. 2005]. Το Double

Warping μετατρέπει στην πραγματικότητα το πρόβλημα μείωσης του latency στο δίκτυο σε πρόβλημα επιλογής αναφοράς βάσει περιεχομένου.

Κατηγορίες	Τεχνικές
Local rendering	Αποθήκευση τρισδιάστατων δεδομένων στον client. Μετάδοση meshes στον client και εφαρμογή του local rendering.
Pre-fetch	Προ-αποστολή
Image Based Rendering	3D image warping. Double warping

Πίνακας 3: Τεχνικές βελτιστοποίησης Latency

2.1.5 Μετάδοση δεδομένων (Data transmission)

Στα συστήματα remote rendering, τα δεδομένα που μεταφέρονται από server σε client συμπιέζονται χρησιμοποιώντας διάφορους αλγόριθμους (codec) προκειμένου να μειωθεί ο φόρτος δικτύου. Υπάρχουν πολλά εργαλεία που μπορούν να χρησιμοποιηθούν για τη συμπίεση εικόνων, βίντεο, και γραφικών, συμπεριλαμβανομένης της συμπίεσης χωρίς απώλειες, της συμπίεσης εικόνας/βίντεο με απώλειες, κωδικοποίηση βίντεο σε πραγματικό χρόνο, κωδικοποίηση σύνθετης εικόνας και κωδικοποίηση γραφικών.

Τα εργαλεία συμπίεσης χωρίς απώλειες, όπως τα LZO [Oberhumer 1996], BZIP [Seward 1996],και [Weinberger et al. 1996], συλλέγουν τις επαναλήψεις υπό την μορφή raw data για κωδικοποιημένα δεδομένα με μειωμένο τον ρυθμό τον bit ανά δευτερόλεπτο (bit rate). Η αποσυμπίεση αυτών των κωδικοποιημένων δεδομένων οδηγεί σε τέλεια ανακατασκευή πανομοιότυπων με τα εισερχόμενα raw data. Τα εργαλεία συμπίεσης χωρίς απώλειες είναι γενικής χρήσης και μπορούν να χρησιμοποιηθούν για τη συμπίεση κειμένων, εκτελέσιμων αρχείων, δυαδικών αρχείων, ήχου, εικόνων και γραφικών. Αυτά τα εργαλεία συμπίεσης χωρίς απώλειες χρησιμοποιήθηκαν από τους servers στα αρχικά συστήματα remote rendering [Ma and Camp 2000; Οι Hege et al. 2000; Levoy 1995] για να συμπιέσουν τις rendered εικόνες πριν μεταφερθούν στον client.

Σε σύγκριση με τα εργαλεία συμπίεσης χωρίς απώλειες, τα εργαλεία συμπίεσης εικόνας με απώλειες αφαιρούν τις λεπτομέρειες που δεν μπορούν να παρατηρηθούν από το ανθρώπινο μάτι για να μειώσουν περαιτέρω το bit rate. Έχουν γίνει πολλές πρώιμες προσπάθειες με την χρήση εργαλείων συμπίεσης εικόνας με απώλειες για τη μόχλευση χαρακτηριστικών ανθρώπινων οπτικών συστημάτων [Ikonomopoulos and Kunt 1985; Kunt et al. 1987; Egger et al. 1995; Campbell et al. 1986; Marcellin et al. 2000; Skodras et al. 2001; Du and Fowler 2007]. Οι υβριδικοί κωδικοποιητές των βίντεο [Wang et al. 2001] ενσωματώνουν εργαλεία συμπίεσης εικόνας με απώλειες με εργαλεία motion-compensation για την εκμετάλλευση του χρονικού πλεονασμού. Οι

υβριδικοί κωδικοποιητές βίντεο έχουν τυποποιηθεί ως, π.χ., MPEG-4 [Schafer 1998] και H.264/AVC [Wiegand et al. 2003], οι οποίοι χρησιμοποιούνται από πολλά σύγχρονα συστήματα remote rendering [Noimark and Cohen-Or 2003; Lamberti και Sanna 2007; Jurgelionis et al. 2009; De Winter et al. 2006; Perlman et al. 2010; Shi et al. 2011a; Herzog et al. 2008; Huang et al. 2013]. Όταν απαιτείται εξαιρετικά χαμηλή καθυστέρηση κωδικοποίησης, της τάξης των νανοδευτερόλεπτων, κωδικοποιητές βίντεο intra-frame [Lee and Song 2008; Nadeem et al. 2010], οι οποίοι συμπιέζουν μεμονωμένα video frames, μπορούν να χρησιμοποιηθούν.

Τα εργαλεία κωδικοποίησης σύνθετων εικόνων στοχεύουν στη συμπίεση υπολογιστικών επιφανειών που αποτελούνται από λιγότερες κινήσεις, αλλά ένα συνδυασμό κειμένων, γραφικών και φυσικών εικόνων. Το Shape Primitive Extraction and Coding (SPEC) [Lin and Hao 2005] τμηματοποιεί κάθε εικόνα σε κείμενα/γραφικά και περιοχές φυσικών εικόνων, κωδικοποιεί τα κείμενα/περιοχές γραφικών χρησιμοποιώντας έναν αλγόριθμο συμπίεσης χωρίς απώλειες και τις φυσικές περιοχές χρησιμοποιώντας JPEG. Μια παρόμοια ιδέα προτάθηκε από τους Maheswari και Radha το [2010]. Οι Wang και Lin [2009] πρότειναν ταυτόχρονη συμπίεση κάθε μακρομπλοκ χρησιμοποιώντας H.264 [Wiegand et al. 2003] και gzip [Gailly 1992] και επιλέγεται το καλύτερο συμπιεσμένο macroblock με τον τρόπο παραμόρφωσης ρυθμού (rate-distortion). Αργότερα πρότειναν ένα γενικευμένο σύστημα που ονομάστηκε United Coding (UC) [Wang και Lin 2012] που βασίζεται σε πολλαπλούς κωδικοποιητές χωρίς απώλειες, όπως ο Run-Length Coding (RLE), gzip, και Portable Network Graphics (PNG).

Τα εργαλεία κωδικοποίησης γραφικών χρησιμοποιούνται για τη συμπίεση γεωμετρικών δεδομένων γραφικών, π.χ. πολυγωνικά πλέγματα (polygonal meshes), νέφη σημείων (point clouds) και δεδομένα όγκου (volume data). Η συμπίεση polygonal meshes εξετάστηκε σε μια πρώιμη μελέτη συμπίεσης γεωμετρίας [Deering 1995], η οποία δημιούργησε μια γραμμική ροή polygonal meshes πριν από την εφαρμογή κωδικοποίησης Huffman, delta και VLC. [Li και Kuo 1998], τα polygonal meshes απεικονίσθηκαν με τοπολογικά και γεωμετρικά δεδομένα και διαφορετικά εργαλεία κωδικοποίησης εφαρμόστηκαν σε αυτά. Η τοπική συμπίεση των polygonal meshes προτάθηκε για αλγόριθμους συμπίεσης σε mesh χαμηλής πολυπλοκότητας [Gumhold και StraBer 1998].

Οι εικόνες βάθους παίζουν σημαντικό ρόλο στο image base rendering. Η πρόβλεψη σχήματος επιφάνειας χρησιμοποιήθηκε για τη συμπίεση ορισμένων περιοχών [Zanuttigh και Cortelazzo 2009]. Το LDI (Layered Depth Image) έχει χρησιμοποιηθεί σε ορισμένα συστήματα remote rendering [Chang and Ger 2002], τα οποία μπορούν να συμπιεστούν με αλγόριθμους. Μελέτες [Milani and Calvagno 2010; Shi et al. 2011a; Oh and Ho 2006; Morvan et al. 2007] έδειξαν ότι οι τροποποιημένοι κωδικοποιητές H.264 [Wiegand et al. 2003] μπορούν να κωδικοποιήσουν αποτελεσματικά εικόνες βάθους.

Σε ένα σύστημα remote rendering, ο κωδικοποιητής λειτουργεί συνήθως στον ίδιο server ως μηχανή απόδοσης (rendering engine). Οι ερευνητές έχουν βρει ότι καλύτερη επίδοση κωδικοποίησης μπορεί να επιτευχθεί με το jointly encoding των rendered images με τα μεταδεδομένα (meta data) να έχουν αφαιρεθεί από την rendering engine. Οι Noimark και Cohen-Or [2003] χρησιμοποίησαν πληροφορίες

γραφικών για τμηματοποίηση της πίσω σκηνής (background) και του προσκηνίου (foreground), εφάρμοσαν διαφορετικές παραμέτρους κβαντισμού, ανίχνευσαν τομές σκηνής (scene cut) χρησιμοποιώντας πληροφορίες κίνησης και ανέλυσαν την οπτική ροή ώστε να αυξήσει την ταχύτητα της κίνησης. Όλα αυτά τα βήματα συμβάλλουν στην αύξηση επίδοσης της κωδικοποιήσης MPEG-4. Pajak et al. [2011] πρότεινε τη συμπίεση εικόνων βάθους χαμηλής ανάλυσης χρησιμοποιώντας H.264 [Wiegand et al. 2003] και ροή των βίντεο H.264 μαζί με αύξηση πληροφορίες όπως ακμές και κινήσεις. Κατά την παραλαβή, στις εικόνες βάθους αυξάνεται ο αριθμός τον pixel (upscaled) στην αρχική τους ανάλυση με τη βοήθεια ακμών και κινήσεων.

Ροή δεδομένων (Data streaming)

Πολλά πρωτόκολλα δικτύου έχουν προταθεί για τη ροή συμπιεσμένων βίντεο μέσω διαδικτύου. Για παράδειγμα, το GLX [Phil Karlton 2005] σχεδιάστηκε ως επέκταση του πρωτοκόλλου X11 για λειτουργίες 3D streaming rendering και το NX [BerliOS 2008] προτάθηκε για τη βελτίωση της απόδοσης στο remote rendering του X11/GLX. Η συμπίεση, η προσωρινή αποθήκευση (caching) , και η καταστολή χρόνου του round trip περιλαμβάνονται στο πρωτόκολλο NX για την ενίσχυση της συνολικής ταχύτητας ροής λειτουργιών Xlib και OpenGL. Το Remote Frame Buffer (RFB) είναι ένα πρωτόκολλο ανοιχτού δικτύου που χρησιμοποιείται από συστήματα VNC [Richardson et al. 1998] για την πραγματοποίηση stream μιας ενημέρωσης του frame buffer. ThiNC, [Baratto et al. 2005], είναι ένα άλλο πρωτόκολλο που μεταδίδει λειτουργίες σχεδίασης χαμηλού επιπέδου. Τα πρωτόκολλα δικτύου που προτείνονται για video streaming [Li et al. 2013], όπως το RTP και το DASH, μπορούν επίσης να χρησιμοποιηθούν σε συστήματα remote rendering. 3TP για γραφικά [AlRegib και Altunbasak 2005] είναι ένα πρωτόκολλο δικτύου για ροή τρισδιάστατων μοντέλων σε κανάλια με απώλειες πάνω από το TCP και το UDP.

Αξιολόγηση QoS/QoE

Ποιότητα υπηρεσίας (QoS) είναι η περιγραφή ή η μέτρηση συνολικής απόδοσης μιας υπηρεσίας ενός δικτύου υπολογιστών. Για την ποσοτική μέτρηση του QoS λαμβάνονται υπόψη πολλές πτυχές σχετικές με την υπηρεσία του δικτύου όπως απώλεια πακέτων δεδομένων, bit rate, throughput, καθυστέρηση μετάδοσης. Η ποιότητα εμπειρίας (QoE) είναι όρος που χρησιμοποιείται για να περιγράψει τη συνολική ικανοποίηση χρήστη όταν χρησιμοποιεί ένα δίκτυο υπολογιστών. Για την μέτρηση QoE μπορούν να χρησιμοποιηθούν ανθρώπινες αξιολογήσεις από ερωτηματολόγια.

Ο χώρος σχεδιασμού της συμπίεσης δεδομένων και της ροής δεδομένων σε συστήματα remote rendering είναι μεγάλος. Επομένως, οι τεχνικές μέτρησης QoS και QoE για συστήματα remote rendering είναι σημαντικές για την αξιολόγηση του συστήματος. Ο Serral-Gracia et al. [2010] ερεύνησε την μέτρηση QoE των συστημάτων video streaming. Ο Stegmaier et al. [2003] συζήτησε σε βάθος πώς οι παράμετροι του QoS επηρεάζουν το remote visualization. Ο Paravati et al. το [2011] μελέτησε τις μετρήσεις QoS και QoE και την παρακολούθηση της επίδοσης για το απομακρυσμένο rendering VE σε κινητές συσκευές. Οι Wang και Dey [2009] πρότειναν ένα μοντέλο

QoE για cloud gaming για κινητά, το οποίο λαμβάνει υπόψη τον τύπο του παιχνιδιού, την ανάλυση, το frame rate, την ποιότητα βίντεο, την καθυστέρηση και την απώλεια πακέτων. VQ (Ποιότητα βίντεο) [Nieh et al. 2003] είναι μια άλλη μέτρηση η οποία χρησιμοποιείται για την μέτρηση της επίδοσης του video streaming ενός συστήματος thin-client. Λόγω της απαίτησης frame rate για την αναπαραγωγή βίντεο, το σύστημα μπορεί να ρίχνει ενεργά video frame εάν φτάσουν αργά.

2.2 Μηχανές παιχνιδιών (Game engines)

Οι σημερινοί δημιουργοί παιχνιδιών βασίζονται σε game engines για την ανάπτυξη των κύριων κομματιών λογισμικού για τα παιχνίδια τους. Μια μηχανή παιχνιδιών απλοποιεί την δουλειά των προγραμματιστών καθώς διάφορες διεργασίες του hardware και του λειτουργικού συστήματος (πάνω στα οποία τρέχει το παιχνίδι) έχουν ήδη πραγματοποιηθεί από τον δημιουργό της game engine. Ο σκοπός μιας game engine είναι επίσης να εκμεταλλευτεί πλήρως τις δυνατότητες των συσκευών έτσι ώστε ο παίκτης να έχει την πιο καθηλωτική εμπειρία παιχνιδιού. Ευρύτερα διαχωρίζουμε την game engine ως framework για τους δημιουργούς παιχνιδιών και την game engine ως ένα κομμάτι κώδικα για τους παίκτες.

Από τη μία πλευρά, η game engine είναι το σύνολο των εργαλείων (συμπεριλαμβανομένων των βιβλιοθηκών χαμηλού επιπέδου, User interface editor και εργαλεία διαχείρισης πολυμέσων του παιχνιδιού) που διευκολύνουν την δουλειά του προγραμματιστή παιχνιδιών στη διαδικασία δημιουργίας ενός νέου παιχνιδιού. Επομένως η κοινότητα των προγραμματιστών παιχνιδιών θεωρεί επομένως μια game engine ως framework ή πλατφόρμα. Το framework παρέχει ένα επίπεδο μεταξύ του περιεχομένου του παιχνιδιού (περιεχόμενο πολυμέσων και κύρια σενάρια) και το υποκείμενο hardware. Τα πιο δημοφιλή framework όπως η Unity3D είναι cross-platform, δηλαδή η game engine μπορεί να λειτουργήσει σε διάφορα λειτουργικά συστήματα και διάφορα διαφορετικά hardware.

Από την άλλη πλευρά, η game engine είναι το σύνολο των λογισμικών και δεδομένων που τελικά εκτελούνται σε μια συσκευή για την παροχή του παιχνιδιού στον χρήστη. Κατά συνέπεια η κοινότητα των παικτών θεωρεί μια game engine ως κομμάτι κώδικα. Όλα τα παιχνίδια που έχουν δημιουργηθεί από το ίδιο framework μοιράζονται ομοιότητες.

2.2.1 Βασικά modules

Μια game engine αποτελείται από διάφορα modules. Για ένα παιχνίδι, μερικά από αυτά είναι γραμμένα κυρίως από τον δημιουργό του παιχνιδιού:

Τεχνητή Νοημοσύνη (AI) – η διαχείριση των non-player character (NPC) επιτυγχάνεται από ορισμένα συγκεκριμένα modules, που συνδυάζουν προ-υπολογισμένο σενάριο με συμπεριφορές που δημιουργούνται εκείνη την στιγμή για να δώσουν την ψευδαίσθηση της νοημοσύνης.

Μηχανή φυσικής προσομοίωσης (physics engine) – αυτά τα modules στοχεύουν να κάνουν τον κόσμο του παιχνιδιού όσο το δυνατόν πιο ρεαλιστικό ανεξάρτητα από τα γεγονότα που αποφασίζονται από τον παίκτη (για τον κεντρικό χαρακτήρα) και από τα AI modules (για τα NPC). Αυτά τα modules προσομοιώνουν τις συμπεριφορές κινητών στοιχείων του κόσμου σύμφωνα με τους νόμους της φυσικής.

Scripting – αυτά τα modules περιέχουν το gameplay. Από τα καταγεγραμμένα input, ο προγραμματιστής παιχνιδιών περιγράφει λεπτομερώς το περιεχόμενο παιχνιδιού και τα events σε μια scripting γλώσσα, η οποία είναι συγκεκριμένη για κάθε framework.

Ορισμένα άλλα modules είναι συνήθως κοινά μεταξύ όλων των παιχνιδιών που δημιουργήθηκαν με ένα συγκεκριμένο framework. Αυτά τα modules αντιπροσωπεύουν το abstraction layer και εμποδίζουν τους δημιουργούς παιχνιδιών να ξοδέψουν χρόνο για θέματα χαμηλού επιπέδου. Συγκεκριμένα:

Input – η λήψη των εντολών από τον παίκτη (π.χ. το joystick, το πληκτρολόγιο και οι αισθητήρες VR).

Απόδοση πολυμέσων (Multimedia rendering) – αυτά τα module είναι υπεύθυνα για την δημιουργία γραφικών και ηχητικών στοιχειών του παιχνιδιού. Όσον αφορά τα γραφικά, τα rendering modules πρέπει να δημιουργούν ένα νέο frame κάθε x χιλιοστά του δευτερολέπτου (ms) όπου x είναι συνήθως μεταξύ 10 και 30. Επίσης η έξοδος από αυτές τις ενότητες είναι συνήθως ακατέργαστο βίντεο, το οποίο πρέπει να κωδικοποιηθεί για την εμφάνιση σε οθόνη εξ αποστάσεως.

Networking – αυτά τα modules εφαρμόζουν ρουτίνες επικοινωνίας και πρωτόκολλα για παιχνίδια για πολλούς παίκτες και παιχνίδια που βασίζονται σε server. Όσον αφορά τις απαιτήσεις των παικτών, τα modules δικτύωσης θα πρέπει να είναι γρήγορα και ανθεκτικά σε κακόβουλα λογισμικά.

2.3 Remote rendering σε ασύρματο VR

Η «Εικονική Πραγματικότητα» (VR) είναι μια τεχνητή απόδοση περιβάλλοντος που χρησιμοποιεί headset και οπτικά πεδία πιθανώς συμπληρωμένα με αισθητηριακές συσκευές.

2.3.1 Απαιτήσεις QoE για εφαρμογές εικονικής πραγματικότητας

Η υποστήριξη σε VR είναι πιο δύσκολη από την υποστήριξη 360 βίντεο. Εκτός από ένα headset και έναν renderer, ένα σύστημα VR αποτελείται από έναν τηλεχειριστήριο (controller) που λαμβάνει αλληλεπιδράσεις από τον χρήστη μέσω φυσικών κουμπιών και αισθητήρων που ενεργοποιούν τη διαδραστικότητα του χρήστη με τον εικονικό κόσμο. Ενώ οι απαιτήσεις QoE για αποδεκτή εμπειρία χρήστη σε συστήματα VR εμφανίζονται με τα βίντεο 360, συγκεκριμένα:

(1) Χαμηλή καθυστέρηση εισόδου στην οθόνη, δηλαδή κάτω από 10-25 ms

(2) Οπτικά εφέ υψηλής ποιότητας, δηλαδή, υποστήριξη frames ανάλυσης 4K ή υψηλότερης

(3) Κινητικότητα, δηλαδή, το headset ή το σύστημα VR θα πρέπει να είναι ασύρματο ώστε να μην περιορίζονται οι αλληλεπιδράσεις του χρήστη.

Η υποστήριξη για αλληλεπιδράσεις του χρήστη σε VR μέσω του controller καθιστά αυτές απαιτήσεις QoE πιο δύσκολο να επιτευχθούν. Αυτό είναι επειδή σε αντίθεση με το video streaming 360 μοιρών όπου η κίνηση του κεφαλιού μπορεί να προβλεφθεί ώστε να εκτελέσει prefetch σε video frames, ενώ στην περίπτωση του VR οι αλληλεπιδράσεις του χρήστη μέσω controller είναι δύσκολο να προβλεφθούν.

Τα υπάρχοντα συστήματα VR μπορούν να χωριστούν σε δύο κατηγορίες: VR υψηλής ποιότητας και αυτόνομα συστήματα VR. Λόγω των απαιτήσεων υψηλής ποιότητας και χαμηλού latency, τα περισσότερα συστήματα VR υψηλής ποιότητας, όπως το HTC Vive και το Oculus Rift, αξιοποιούν έναν ισχυρό επιτραπέζιο υπολογιστή H/Y για rendering πλούσιου περιεχομένου γραφικών σε υψηλά frame rate και υψηλή οπτική ποιότητα. Ωστόσο, περισσότερες από αυτές τις λύσεις πρέπει να συνδεθεί σε υπολογιστή μέσω καλωδίου USB για αποστολή δεδομένων αισθητήρα από το Head Mounted Display (HMD) στον υπολογιστή και ένα καλώδιο HDMI για την αποστολή γραφικών από τον υπολογιστή πίσω στο HMD. Αυτά τα καλώδια όχι μόνο περιορίζουν την κινητικότητα του χρήστη αλλά επιβάλλουν και κινδύνους όπως ο χρήστης να σκοντάψει στο καλώδιο. Αυτόνομα, φορητά συστήματα VR όπως το Samsung Gear VR και Το Google Daydream εκτελούν εφαρμογές VR και πραγματοποιούν rendering γραφικών τοπικά στο VR headset (ή σε ένα smartphone στο headset). Αυτά τα συστήματα VR επιτρέπουν την ασύρματη χρήση, αλλά η ποιότητα του rendering είναι περιορισμένη από τη δυνατότητα του headset ή του smartphone.

Τα ασύρματα συστήματα VR υψηλής ποιότητας είναι ιδιαίτερα επιθυμητά αλλά εξαιρετικά δύσκολα. Στην ιδανική περίπτωση, τα καλώδια μεταξύ του VR HMD και του υπολογιστή θα έπρεπε να αντικατασταθούν από ασύρματη σύνδεση. Ωστόσο, ακόμη και τα υπάρχοντα υψηλής ποιότητας συστήματα VR λειτουργούν σε ανάλυση 2880 x

2720 και 90 Hz, γεγονός που παράγει ρυθμό μετάδοσης δεδομένων πολύ υψηλότερο από εκείνους που υποστηρίζονται από τα υπάρχοντα προϊόντα ασύρματης επικοινωνίας όπως Wi-Fi και ασύρματη επικοινωνία 60 GHz. Η απαραίτητη κωδικοποίηση εικόνας καθιστά δύσκολη τη διατήρηση των αυστηρών απαιτήσεων motion to photon latency (η καθυστέρηση από την στιγμή που ο χρήστης κινείται μέχρι να εμφανιστεί στην οθόνη) του VR, οι οποίες είναι απαραίτητες για τη μείωση το motion sickness.

Οι περισσότερες υπάρχουσες έρευνες έχουν επικεντρωθεί στη βελτιστοποίηση μίας ασύρματης σύνδεσης ή του pipeline γραφικών VR. Συστήματα όπως το TPCAST και το DisplayLink αντικαθιστούν το καλώδιο HDMI με μια ασύρματη σύνδεση για να ενεργοποιήσουν την ασύρματη VR εμπειρία με remote rendering και streaming. Ωστόσο, καμία εξ αυτών μελέτησαν τρόπους βελτιστοποίησης που προκύπτουν κατά τον συνδυασμό rendering, streaming και display. Επιπλέον, εξακολουθεί να είναι δύσκολο να υποστηρίξει ασύρματο VR για 4K ή 8K συστήματα με framerate μεγαλύτερα από 90 Hz. Επιπρόσθετα, η εμφάνιση remote rendering frames σε ένα HMD μπορεί εισάγει επιπλέον latency λόγω του rendering που βασίζεται στο Vsync.

2.4 Μέθοδοι remote rendering σε VR

Σε αυτή την ενότητα θα αναλύσουμε δύο μεθόδους remote rendering που υλοποιήθηκαν ώστε να καταλάβουμε με ποιον τρόπο ξεπεράστηκαν οι παραπάνω προκλήσεις.

2.4.1 Μέθοδος Liu

Η μέθοδος του Liu [Liu et al 2018] αναφέρεται σε μια ανοιχτή πλατφόρμα remote rendering που μπορεί να υποστηρίξει υψηλής ποιότητας ασύρματο VR με χαμηλό latency σε υπολογιστή με hardware για γενική χρήση. Μειώνει το latency του streaming που προκαλείται από το rendering των frames, την κωδικοποίηση, τη μετάδοση, την αποκωδικοποίηση και την εμφάνιση μέσω Parallel Rendering and Streaming Pipeline (PRS) και Remote Vsync Driven Frame Rendering (RVDR). Το PRS διοχετεύει το rendering, την κωδικοποίηση, διαδικασία μετάδοσης και αποκωδικοποίησης. παραλληλίζει την διαδικασία κωδικοποίησης των frames σε κωδικοποιητές GPU. Η τεχνική RVDR προγραμματίζει προσεκτικά την ώρα έναρξης του αισθητήρα και του rendering νέων frames στην πλευρά του server, έτσι ώστε το αποτέλεσμα να φτάσει στην οθόνη πριν από το σήμα Vsync, μειώνοντας έτσι το latency που προκαλείται από την ενημέρωση της οθόνης.

2.4.1.1 Προκλήσεις και ανάλυση του latency

Ο σχεδιασμός ενός συστήματος υψηλής ποιότητας ασύρματου VR είναι εξαιρετικά δύσκολος λόγω των αυστηρών απαιτήσεων σχετικά με το throughput δεδομένων και την latency μεταξύ των συσκευών. Υποθέτοντας ότι χρησιμοποιεί τρία byte για την κωδικοποίηση δεδομένων RGB του κάθε pixel, για HTC Vive και Oculus Rift με framerate 90Hz και ανάλυση 2160x1200, ο ρυθμός μετάδοσης ακατέργαστων δεδομένων είναι 5,6 Gbps, πολύ υψηλότερο από τον ρυθμό μετάδοσης δεδομένων (π.χ. λιγότερο από 2 Gbps) που υποστηρίζεται από υπάρχοντα προϊόντα ασύρματης επικοινωνίας όπως Wi-Fi και ασύρματη επικοινωνία 60 GHz. Για VR σε ανάλυση 4K UHD ή ακόμα και 8K UHD, ο απαιτούμενος ρυθμός μετάδοσης δεδομένων θα έφτανε τα 17,9 Gbps και 71,7 Gbps, αντίστοιχα.

Για να αντιμετωπιστεί η πρόκληση του υψηλού throughput, είναι απαραίτητη η συμπίεση δεδομένων. Ωστόσο, η υψηλής ποιότητας VR απαιτεί ένα αυστηρά χαμηλό συνολικό motion to photon latency 20- 25 ms για την μείωση του motion sickness. Δηλαδή, μόλις κινηθεί το HMD, το σύστημα πρέπει να μπορεί να εμφανίζει ένα νέο frame που δημιουργείται από την νέα στάση του HMD σε 20-25 ms. Καθώς η συμπίεση και η αποσυμπίεση των frames εισάγουν επιπλέον latency, είναι ακόμη πιο δύσκολο να καλυφθούν οι απαιτήσεις του latency μεταξύ των συσκευών.

Για την ανάλυση του latency του προτεινόμενου συστήματος ασύρματου VR με remote rendering χρησιμοποιήθηκαν οι παρακάτω εξισώσεις:

$$1)\ T_{e2e}=T_{sence}+T_{render}+T_{stream}+T_{display}$$

$$2)\ T_{stream}=T_{encode}+T_{trans}+T_{decode}$$

$$3)\ T_{trans}=\frac{FrameSize}{Throughput}$$

Το T_{e2e} είναι η συνολική latency μεταξύ των συσκευών στην δημιουργία και την εμφάνιση ενός νέου frame. Αποτελείται από τέσσερα μέρη: τον χρόνο του rendering server για ανάκτηση δεδομένων του αισθητήρα από το HMD (T_{sence}). Τον χρόνο για να δημιουργήσει ο rendering server ένα νέο frame (T_{render}). Τον χρόνο για αποστολή του νέου frame από τον rendering server στο HMD (T_{stream}) και τον χρόνο για να εμφανίσει το HMD το νέο frame ($T_{display}$).

Το T_{stream} είναι επιπλέον latency που εισάγεται σε ασύρματο VR. Έχει τρία μέρη: το χρόνο συμπίεσης του frame στον rendering server (T_{encode}). Ο χρόνος για τη μετάδοση του συμπιεσμένου frame από τον rendering server στο HMD μέσω ασύρματης σύνδεσης (T_{trans}). Και ο χρόνος αποσυμπίεσης του frame στο HMD (T_{decode}). Ο χρόνος T_{trans} αποφασίζεται από το μέγεθος του συμπιεσμένου frame και το throughput δεδομένων στην ασύρματη σύνδεση.

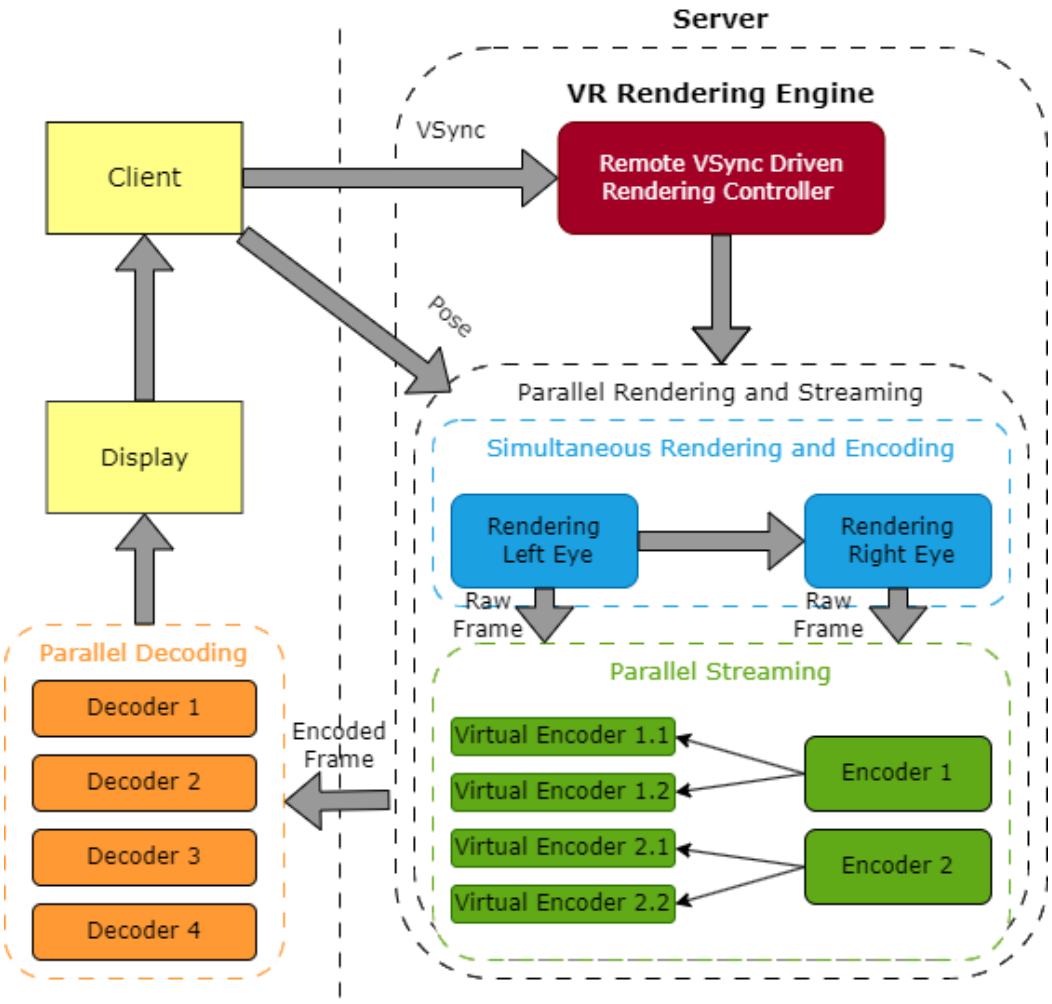
Το $T_{display}$ εισάγει εξίσου σημαντικό latency. Στα σύγχρονα συστήματα γραφικών, η εμφάνιση των frame καθοδηγείται από σήματα Vsync που παράγονται περιοδικά με βάση τον ρυθμό ανανέωσης της οθόνης. Αν ένα frame χάσει το τρέχον σήμα Vsync αφού ληφθεί και αποκωδικοποιηθεί σε ένα HMD, πρέπει να περιμένει στο frame buffer

για το επόμενο σήμα Vsync προτού μπορέσει να εμφανιστεί στην οθόνη Για framerate 90 Hz, ο μέσος χρόνος αναμονής είναι 5,5 ms. Μια τέτοια επιπλέον latency μπορεί να επηρεάσει σημαντικά την επίδοση ενός συστήματος υψηλής ποιότητας ασύρματου VR, και επομένως πρέπει να περιοριστεί όσο το δυνατόν περισσότερο.

Για τον συνολικό 20-25 ms του T_{e2e}, το T_{sence} είναι μικρό (λιγότερο από 400μs στο σύστημά με ένα δίκτυο WiGig[1]). Το T_{render} μπορεί να είναι 5-11ms ανάλογα με το φορτίο του rendering. Με $T_{display}$ των 5,5ms, απομένουν λιγότερα από 10 ms για το T_{stream}, συμπεριλαμβανομένης της κωδικοποίησης, της μετάδοσης και την αποκωδικοποίηση ενός frame, κάτι που το καθιστά πολύ δύσκολο να επιτευχθεί η απαιτούμενη latency για VR υψηλής ποιότητας.

[1] Το WiGig είναι ένα σύνολο πρωτοκόλλων ασύρματου δικτύου 60GHz

2.4.1.2 System Overview



Εικόνα 2: Αρχιτεκτονική Συστήματος

(Πηγή: https://www.researchgate.net/publication/326240756_Cutting_the_Cord_Designing_a_High-quality_Untethered_VR_System_with_Low_Latency_Remote_Rendering Liu et al [2018])

Το Σχήμα στην εικόνα 2 δείχνει την αρχιτεκτονική του προτεινόμενου συστήματος ασύρματου VR με remote rendering. Σε υψηλό επίπεδο, έχει δύο μέρη συνδεδεμένα μέσω ασύρματης σύνδεσης: ένα HMD ως client και έναν υπολογιστή ως rendering server. Ο HMD client καταγράφει τη στάση του παίκτη και το timing του σήματος Vsync και στέλνει τα καταγεγραμμένα δεδομένα στο rendering server. Εάν ο παίκτης χρησιμοποιεί επιπλέον χειριστήρια για να παίξει το παιχνίδι, ο client στέλνει επίσης τα δεδομένα του χειριστηρίου στον server. Χρησιμοποιώντας τα δεδομένα που λαμβάνονται από τον client, ο server κάνει render νέα frame, τα συμπιέζει και τα μεταδίδει στον client. Με την παραλαβή ενός νέου frame, ο client το αποσυμπιέζει και

το εμφανίζει στο HMD. Η ασύρματη σύνδεση μπορεί να είναι WiFi ή ασύρματη επικοινωνία 60 GHz ως WiGig.

Για να μειωθεί το latency των streaming frames από τον rendering server στο HMD client, αναπτύχθηκαν δύο βασικές τεχνικές. Η πρώτη τεχνική είναι η παράλληλη απόδοση και ροή PRS (Parallel Rendering and Streaming). Το PRS παίρνει πλεονέκτημα του rendering δυο εικόνων σε VR (μια για το κάθε μάτι). Μόλις η εικόνα του αριστερού ματιού γίνεται rendered, το PRS τη στέλνει αμέσως στον κωδικοποιητή για συμπίεση. Ταυτόχρονα, το PRS συνεχίζει να κάνει render την εικόνα του δεξιού ματιού, επιτρέποντας το ταυτόχρονο rendering και την κωδικοποίηση. Το PRS διαιρεί περαιτέρω την εικόνα κάθε ματιού σε δύο διαφάνειες για παράλληλη κωδικοποίηση τεσσάρων κατευθύνσεων για την πλήρη αξιοποίηση των δυνατότητων του hardware encoding. Αφού κωδικοποιηθεί μια διαφάνεια πλαισίου, αποστέλλεται αμέσως στην ασύρματη σύνδεση για μετάδοση, χωρίς να περιμένει να είναι όλο το frame συμπιεσμένο. Ομοίως, μόλις το HMD λάβει μια διαφάνεια πλαισίου αρχίζει αμέσως να το αποσυμπιέζει, χωρίς να περιμένει τις άλλες διαφάνειες πλαισίου. Κατά συνέπεια, επιτυγχάνουμε παράλληλο rendering του frame, κωδικοποίηση, μετάδοση και αποκωδικοποίηση, μειώνοντας έτσι σημαντικά το latency.

Η δεύτερη τεχνική είναι η Remote Vsync Driven Rendering (RVDR). Η βασική ιδέα του RVDR είναι να μειώσει το display latency αποφασίζοντας πότε θα αποκτήσει τα δεδομένα του αισθητήρα κεφαλιού και να κάνει render ένα νέο frame σχετικά με το χρονισμό των σημάτων Vsync του HMD client. Ο client συνεχίζει να ανιχνεύει τον χρόνο του τελευταίου σήματος Vsync και την καθυστέρηση εμφάνισης του τελευταίου frame. Με βάση αυτές τις χρονικές πληροφορίες, ο rendering server αποφασίζει εάν θα ξεκινήσει νωρίτερα την απόδοση του επόμενου frame, ώστε το επόμενο frame να μπορεί να συναντήσει το επόμενο σήμα Vsync στον client, ή για να αναβάλει ελαφρώς την απόκτηση δεδομένων του αισθητήρα και το rendering του επόμενου frame ώστε να φτάσει πιο κοντά στο σήμα Vsync και να μειωθεί η display latency. Αυτό είναι αποτελεσματικό γιατί το frame μπορεί να γίνει render με την πιο πρόσφατη δυνατή στάση του HMD και μειώνει έτσι την motion to photon latency. Ως αποτέλεσμα, συνεχίζουμε να μειώνουμε την καθυστέρηση της οθόνης και ελαχιστοποιείται το ποσοστό των καρέ που λείπουν για μεγιστοποίηση της εμπειρίας του χρήστη.

Στο σχέδιο αυτό, ο HMD client χρησιμοποιεί hardware video codec (π.χ., H.264) για την αποκωδικοποίηση των frame που γίνονται render στον rendering server. Από την συγκεκριμένη εργασία επιλέχθηκε αυτό το σχέδιο επειδή οι hardware video codec έχουν μικρό μέγεθος και καταναλώνουν χαμηλή ισχύ. Προηγούμενες μελέτες [Liu 2017, Lai 2019] έχουν αποδείξει ότι δεν είναι πρακτικά εφικτή η αποκωδικοποίηση των frames με χρήση CPU ή GPU σε HMD, λόγω της υψηλής latency αποκωδικοποίησης και της υψηλής κατανάλωσης ενέργειας. Οι hardware video codec είναι οικονομικά αποδοτικές τεχνικές, που χρησιμοποιούνται ευρέως σε smartphone, tablet, φορητούς υπολογιστές και πολλές άλλες συσκευές. Έτσι, ενσωματώνοντας hardware video codec σε HMD είναι μια πρακτική λύση για την υποστήριξη συστημάτων ασύρματου VR υψηλής ποιότητας .

2.4.1.3 Parallel Rendering and Streaming Pipeline

Το rendering ενός frame με πλούσια γραφικά μπορεί να διαρκέσει πολύ χρόνο (π.χ. μεγαλύτερο από 5 ms) ακόμη και σε μια ισχυρή GPU έτοιμη για VR (π.χ. Nvidia Titan X). Καθώς δεν μπορεί απλώς να μειωθεί η ποιότητα του frame για εξοικονομηθεί χρόνος frame rendering, πρέπει να βρεθούν άλλοι τρόποι για να μετριαστεί η επίδραση του μεγάλου frame rendering χρόνου στο latency μεταξύ των συσκευών. Για το σκοπό αυτό, προτάθηκε η προσέγγιση του ταυτόχρονου rendering και κωδικοποίησης, για να επιτρέψει την έναρξη της κωδικοποίησης ενός frame ενώ γίνεται ακόμα rendered.

Το ταυτόχρονο rendering και κωδικοποίηση είναι εφικτά για δύο λόγους. Πρώτον, παρατηρούμε ότι η απόδοση ενός πλαισίου VR συνήθως γίνεται σε τρία διαδοχικά βήματα:

(1) rendering της εικόνας του αριστερού ματιού

(2) rendering της εικόνας του δεξιού ματιού

(3) εφαρμογή παραμόρφωσης φακού σε ολόκληρο το frame έτσι ώστε το frame να μπορεί να εμφανίζεται σωστά σε ένα VR headset.

Αυτή η διαδοχική απόδοση των εικόνων για δύο μάτια δίνει την ευκαιρία να αρχίσει η κωδικοποίηση για την εικόνα του αριστερού ματιού πριν αποδοθεί η δεξιά εικόνα πλήρως. Δεύτερον, οι σύγχρονες GPU έχουν ειδικούς hardware encoders και decoders που είναι ξεχωριστοί από τους GPU πυρήνες που χρησιμοποιούνται για το rendering των VR frame (π.χ. πυρήνες CUDA σε Nvidia GPU). Επομένως, μπορούν να χρησιμοποιούν οι ειδικοί hardware encoders για να συμπιεστεί ένα frame χωρίς να επηρεαστεί η επίδοση για το VR rendering.

Συγκεκριμένα, για ταυτόχρονο rendering και κωδικοποίηση, επανασχεδιάστηκε η διαδικασία του VR rendering στα ακόλουθα 6 βήματα:

(1) rendering της εικόνας του αριστερού ματιού,

(2) εφαρμογή παραμόρφωσης φακού στο αριστερό μάτι εικόνα

(3) Η παραμορφωμένη εικόνα του αριστερού ματιού περνάει στο pipeline κωδικοποίησης σε ξεχωριστό thread

(4) ταυτόχρονα rendering της εικόνας για δεξιού ματιού

(5) εφαρμογή παραμόρφωσης φακού στην εικόνα του δεξιού ματιού

(6) Η παραμορφωμένη εικόνα του δεξιού ματιού περνάει στο pipeline κωδικοποίησης σε άλλο ξεχωριστό thread.

Μόνο τα βήματα (1), (2), (4) και (5) εκτελούνται στο κύριο rendering thread, ενώ τα βήματα (3) και (6) εκτελούνται σε δύο ξεχωριστά thread κωδικοποίησης χρησιμοποιώντας κωδικοποιητές που βασίζονται σε hardware. Αυτές Οι λειτουργίες κωδικοποίησης καταναλώνουν κυρίως πόρους του κωδικοποιητή που βασίζεται σε

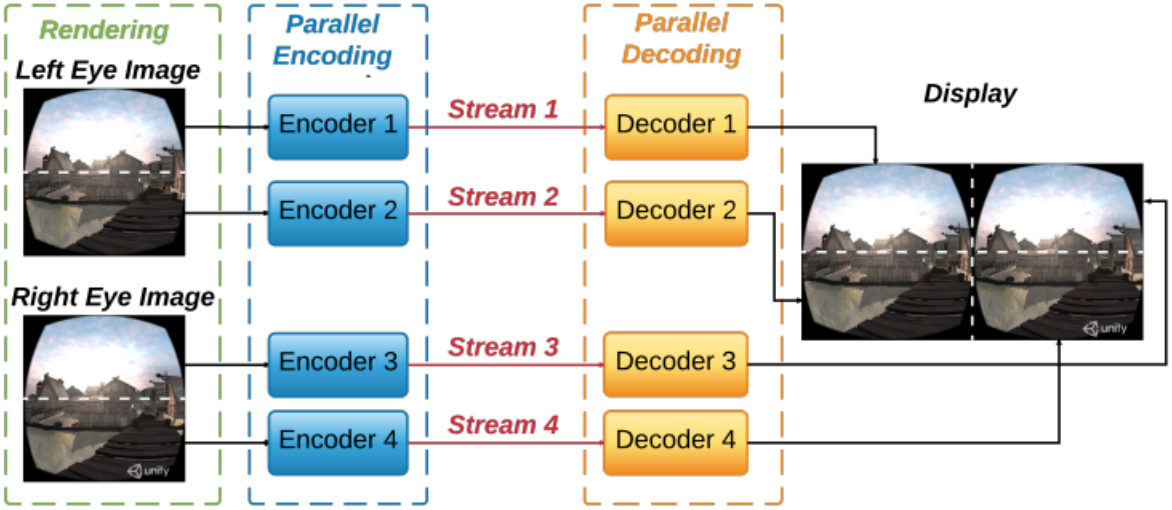
hardware με ελαφριά χρήση της CPU, και επομένως δεν μπλοκάρουν ή επιβραδύνουν το frame rendering pipeline της GPU.

Οι συνήθεις προσεγγίσεις για video streaming συνήθως χρησιμοποιούν κωδικοποιητές βασισμένους σε hardware για να επιταχύνουν τη διαδικασία κωδικοποίησης. Ωστόσο, συνεχίζουν να περιμένουν ένα frame να αποδοθεί πλήρως πριν περάσουν ολόκληρο το frame σε έναν hardware encoder. Ένα τέτοιο σχέδιο αυξάνει σε μεγάλο βαθμό το latency μεταξύ των συσκευών, κάτι που είναι αποδεκτό για video streaming με χαμηλά fps (π.χ. 30 fps) και χωρίς αλληλεπιδράσεις από τον χρήστη, αλλά δεν είναι αποδεκτό σε διαδραστικά συστήματα VR υψηλής ποιότητας

Παράλληλο Streaming

Για περαιτέρω μείωση του streaming latency, προτάθηκε τη χρήση μιας τεχνικής multithreaded streaming για την κωδικοποίηση της εικόνας κάθε ματιού σε πολλαπλά thread κωδικοποίησης. Αυτό οφείλεται στο γεγονός ότι σχεδόν όλες οι GPU υψηλής επίδοσης υποστηρίζουν περισσότερα από ένα session κωδικοποίησης βίντεο και κάθε session κωδικοποίησης δημιουργεί το δικό του stream κωδικοποίησης ανεξάρτητα. Κατά συνέπεια, διαιρώντας μια εικόνα σε διαφάνειες και κωδικοποιώντας κάθε διαφάνεια χρησιμοποιώντας διαφορετικά session κωδικοποίησης παράλληλα, μειώνεται ο συνολικός χρόνος κωδικοποίησης. Στο σύστημα αυτό, η εικόνα από κάθε μάτι κόβεται σε δύο διαφάνειες και η κάθε διαφάνεια συμπιέζεται σε ξεχωριστό video stream. Συνολικά, υπάρχουν τέσσερις διαφάνειες από τα δύο μάτια για 4 κατευθύνσεις παράλληλο streaming. Από την πλευρά του client, πολλαπλά session αποκωδικοποίησης χρησιμοποιούνται για την αποκωδικοποίηση κάθε διαφάνειας από τις εικόνες παράλληλα.

Ο μηχανισμός παράλληλης ροής μπορεί να συνδυαστεί με ταυτόχρονο rendering και κωδικοποίηση. Το σχήμα στην εικόνα 3 δείχνει τη διαδικασία της ταυτόχρονης απόδοσης και κωδικοποίησης μαζί με 4-way παράλληλη ροή. Η εικόνα κάθε ματιού χωρίζεται σε δύο διαφάνειες: το πάνω και το κάτω. Οι συνολικά τέσσερις διαφάνειες εικόνων κωδικοποιούνται σε video stream χρησιμοποιώντας τέσσερις κωδικοποιητές. Αντίστοιχα, Ο HMD client χρησιμοποιεί τέσσερις αποκωδικοποιητές για την αποκωδικοποίηση των τεσσάρων video stream, συνθέτει τις τέσσερις διαφάνειες εικόνας σε πλήρες frame και εμφανίζει το frame στο HMD.

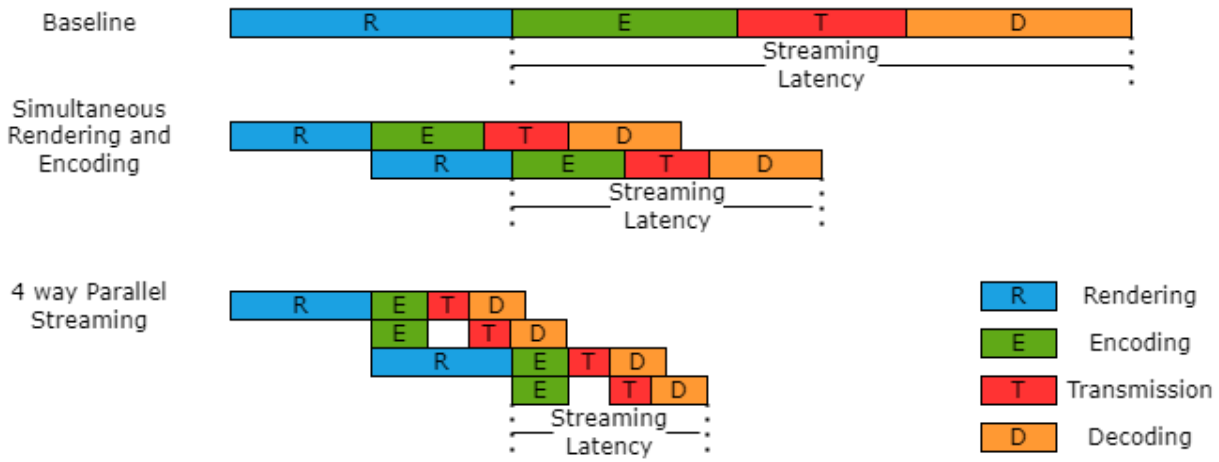


Εικόνα 3: Ταυτόχρονη απόδοση και κωδικοποίηση για παράλληλη μετάδοση

(Πηγή: https://www.researchgate.net/publication/326240756_Cutting_the_Cord_Designing_a_High-quality_Untethered_VR_System_with_Low_Latency_Remote_Rendering Liu et al [2018])

Το σχήμα στην εικόνα 4 δείχνει πώς ο μηχανισμός PRS μπορεί να μειώσει το streaming latency μέσω ταυτόχρονου rendering και κωδικοποίησης και παράλληλο streaming 4 κατευθύνσεων, σε σύγκριση με μια βασική προσέγγιση. Τέσσερις κύριες εργασίες (rendering, encoding, transmission και decoding) απεικονίζονται με ορθογώνια σε διάφορα χρώματα. Το μήκος του κάθε ορθογώνιο είναι ο πρόχειρος χρόνος εκτέλεσης της αντίστοιχης εργασίας. Σημειώστε ότι εδώ αναλύουμε μόνο το streaming latency αντί την συνολική latency μεταξύ των συσκευών. Συνεπώς το σχήμα δεν δείχνει τον χρόνο λήψης των δεδομένων του αισθητήρα πριν από το rendering ενός frame και τον χρόνο που το frame περιμένει στο frame buffer αφού αποκωδικοποιηθεί αλλά πριν εμφανιστεί στην οθόνη. Στην βασική προσέγγιση, οι τέσσερις εργασίες εκτελούνται διαδοχικά. Το streaming latency, που φαίνεται στο Σχήμα 4, είναι ο συνολικός χρόνος εκτέλεσης κωδικοποίησης, μετάδοσης και αποκωδικοποίησης του ολόκληρου frame.

Με ταυτόχρονο rendering και κωδικοποίηση (μέση στο Σχήμα 4), η κωδικοποίηση της εικόνας του αριστερού ματιού ξεκινά αμέσως μετά το rendering της αριστερής εικόνας και παράλληλα με το rendering της εικόνας του δεξιού ματιού. Ως αποτέλεσμα, αυτή η αμφίδρομη παράλληλη προσέγγιση μειώνει το streaming latency κατά ½ αφού το frame γίνει render. Με συνδυασμό το ταυτόχρονο rendering και κωδικοποίηση με παράλληλο streaming 4 κατευθύνσεων μαζί (κάτω στο σχήμα 4), χρησιμοποιούνται δύο κωδικοποιητές για συμπίεση της εικόνας του ενός ματιού παράλληλα. Η πρόσθετη streaming latency μειώνεται περαιτέρω στο ¼ περίπου της βασικής προσέγγισης.



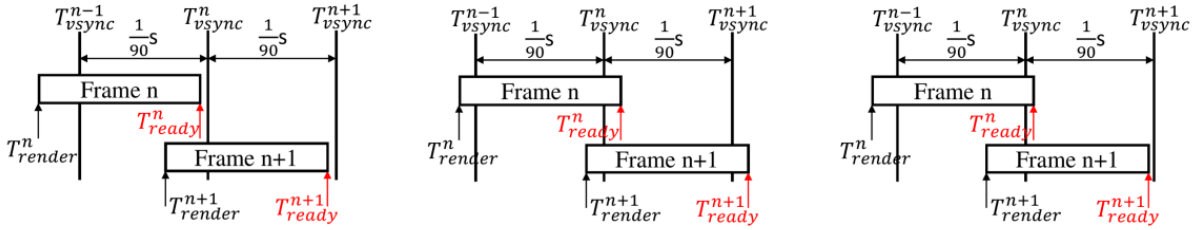
Εικόνα 4: Pipeline της παράλληλης απόδοσης και μετάδοσης

(Πηγή: https://www.researchgate.net/publication/326240756_Cutting_the_Cord_Designing_a_High-quality_Untethered_VR_System_with_Low_Latency_Remote_Rendering Liu et al [2018])

2.4.1.4 Remote Vsync Driven Rendering

Τα σύγχρονα συστήματα υπολογιστών χρησιμοποιούν σήματα Vsync (Κάθετος συγχρονισμός) για συγχρονισμό του ρυθμού των rendering frames (δηλ. ρυθμός καρέ) και του ρυθμού ανανέωσης μιας οθόνης. Για να εξασφαλιστεί μια ομαλή εμπειρία χρήστη (π.χ., αποφύγει του screen tearing), η τεχνική double buffering συνήθως χρησιμοποιείται με δύο frame buffer: ένα μπροστινό frame buffer (front frame buffer) που περιέχει το frame που εμφανίζεται στην οθόνη και ένα πίσω frame buffer (back frame buffer) που περιέχει το frame που γίνεται render. Μετά την παραλαβή ενός σήματος Vsync, το σύστημα αλλάζει τα δύο buffer για να εμφανίσει το νέο frame και συνεχίζει να κάνει render το επόμενο frame στο νέο back frame buffer. Εάν το σύστημα κάνει render ένα frame πολύ γρήγορα, το frame πρέπει να περιμένει για το επόμενο σήμα Vsync στο back frame buffer, πριν σταλεί στην οθόνη. Εάν το rendering ενός frame παίρνει πολύ χρόνο και χάσει το σήμα Vsync, πρέπει να περιμένει για το επόμενο σήμα Vsync που θα εμφανιστεί.

Ο παραπάνω μηχανισμός απόδοσης και εμφάνισης που βασίζεται στο Vsync λειτουργεί καλά σε ένα τοπικό σύστημα. Ωστόσο, στην περίπτωση του remote rendering, το frame που εμφανίζεται καθοδηγείται από σήματα Vsync του HMD client αλλά το frame rendering καθοδηγείται από τα σήματα Vsync του rendering server. Λόγω της ασύγχρονης απόδοσης και εμφάνισης καρέ η επιπλέον streaming latency μπορεί να προκαλέσει προβλήματα.



(a) Ideal case (b) Long waiting time (c) Missing a frame

Εικόνα 5 Εμφάνιση δύο διαδοχικών καρέ που αποδίδονται εξ αποστάσεως

(Πηγή: https://www.researchgate.net/publication/326240756_Cutting_the_Cord_Designing_a_High-quality_Untethered_VR_System_with_Low_Latency_Remote_Rendering Liu et al [2018])

Για να δείξουμε τα προβλήματα, δείχνουμε τη διαδικασία επεξεργασίας δύο διαδοχικών frame n και n+1 στο Σχήμα 5. Ο rendering server αρχίζει να κάνει render το frame n τη στιγμή T^{n}_{render}. Στη συνέχεια, το frame κωδικοποιείται και μεταδίδεται στον client. Ο client αποκωδικοποιεί το frame και το παρουσιάζει στην αλυσίδα εναλλαγής frame buffer στο τη χρονική στιγμή T^{n}_{ready}. Ορίζουμε το χρονικό διάστημα μεταξύ T^{n}_{render} και T^{n}_{ready} (T^{n}_{ready}-T^{n}_{render}), ως τον χρόνο δημιουργίας του frame n. Στην ιδανική περίπτωση, το frame είναι έτοιμο πριν από το σήμα Vsync τη στιγμή T^{n}_{Vsync} , ώστε να μπορεί να εμφανιστεί αμέσως. Αυτή η ιδανική περίπτωση φαίνεται στο Σχήμα 5(α). Ωστόσο, εάν το frame n χάσει το Vsync σήμα n (T^{n}_{ready}>T^{n}_{Vsync}), πρέπει να περιμένει για το Vsync σήμα n+1 και έτσι η latency μεταξύ των συσκευών αυξάνεται σε T^{n+1}_{Vsync}–T^{n}_{ready}. Μια τέτοια περίπτωση μεγάλου χρόνου αναμονής εμφανίζεται στην Εικόνα 5(β). Επιπλέον, εάν το frame n έχασε το Vsync σήμα n,και ταυτόχρονα το frame n+1 είναι έτοιμο πριν από την ώρα T^{n+1}_{Vsync} (T^{n+1}_{ready}<T^{n+1}_{Vsync}), το frame n θα γίνει άχρηστο και έτσι θα πεταχτεί. Αντίθετα, το frame n+1 θα εμφανιστεί στο σήμα Vsync n+1. Αυτή η περίπτωση φαίνεται στο Σχήμα 5(γ). Σε αυτή την περίπτωση, ο χρόνος και οι πόροι που χρησιμοποιούνται για rendering, κωδικοποίηση, μετάδοση και αποκωδικοποίησης του frame n χάνονται.

Για την επίλυση των προβλημάτων, προτάθηκε να οδηγηθεί το frame rendering του server χρησιμοποιώντας τα Vsync σήματα του client. Η βασική ιδέα είναι η προσαρμογή του χρονισμού του rendering του επόμενου frame σύμφωνα με πληροφορίες από τον client για το πώς το προηγούμενο frame εμφανίστηκε, πόση ώρα ήταν ο χρόνος αναμονής του και πόσο γρήγορο το HMD κινήθηκε. Συγκεκριμένα, χρησιμοποιούνται οι παρακάτω εξισώσεις για να αποφασιστεί πότε να ξεκινήσει το rendering ενός νέου frame n+1:

$$T^{n+1}_{render} = T^{n}_{render} + \frac{1}{90}s + T_{shift} \quad (4)$$

$$T_{shift} = (T^{n}_{Vsync} - T^{n}_{ready} - T_{conf} - T_{motion}) * cc \quad (5)$$

$$T_{motion} = k * \Delta\theta^{n} \quad (6)$$

Στην Εξίσωση 4, εκτός από τη χρήση σταθερού χρονικού διαστήματος 1/90 δευτερόλεπτα για να διατηρηθεί το frame rate 90 Hz, εισάγεται περαιτέρω η χρονική μετατόπιση T_{shift} που τροποποιεί δυναμικά την χρονική στιγμή έναρξης του rendering frame n+1 (T_{render}^{n+1}). Το T_{shift} αποφασίζεται από διάφορους παράγοντες. Ο πρώτος παράγοντας είναι ο χρόνος αναμονής του frame n, $T_{Vsync}^{n} - T_{ready}^{n}$. Ας υποθέσουμε ότι τα frame n και n+1 έχουν τον ίδιο χρόνο δημιουργίας, δηλαδή $T_{ready}^{n+1} - T_{render}^{n+1} = T_{ready}^{n} - T_{render}^{n}$, ο χρόνος τοποθέτησης του frame n+1 στην αλυσιδωτή εναλλαγή frame buffer T_{ready}^{n+1} θα είναι ακριβώς ίσο με T_{Vsync}^{n+1}. Με αυτόν τον τρόπο, ελαχιστοποιείται ο χρόνος αναμονής του frame n+1. Ωστόσο, εάν χρειάζεται λίγο μεγαλύτερο χρονικό διάστημα για να δημιουργηθεί το frame n+1, το frame n+1 μπορεί να χάσει το Vsync σήμα n+1, με αποτέλεσμα ένα πολύ μεγάλος χρόνος αναμονής ή ακόμα και να χαθεί το frame n+1. Για να μετριαστεί αυτό το θέμα, εισάγεται το T_{conf} για μετατόπιση του T_{ready}^{n+1}πίσω λίγο για να είναι ανεχτή η διακύμανση στα παραγόμενα frame.

Ακολουθήθηκε μια προσέγγιση με βάση τα δεδομένα για να αποφασιστεί ο σωστός χρόνος T_{conf}. Αρχικά, ορίστηκε το T_{conf} στο μηδέν. Παρακολουθείται ο χρόνος δημιουργίας των frame χρόνου των τελευταίων 1.000 frame. Υπολογίζεται η τιμή που καλύπτουν οι διακυμάνσεις του χρόνου δημιουργίας frame των τελευταίων 1.000 καρέ με διάστημα εμπιστοσύνης 99%. Ορίστηκε η τιμή του T_{conf} στο μισό του εμπιστευτικού διαστήματος εμπιστοσύνης 99%. Με τον χρόνο, το T_{conf} λειτουργεί ως μια προσαρμοστική προστασία για τη διαχείριση της διακύμανσης δημιουργώντας διαδοχικά πλαίσια.

Ένας άλλος παράγοντας που εξετάστηκε στο T_{shift} είναι το πόσο γρήγορα κινείται το HMD. Η διαίσθηση πίσω από αυτή την εξέταση είναι η εξής: όταν το HMD κινείται γρήγορα, η οπτική του VR παιχνιδιού μπορεί να αλλάξει γρήγορα με αποτέλεσμα, το περιεχόμενο του επόμενου frame μπορεί να έχει σημαντικές αλλαγές και επομένως, ο χρόνος για να γίνει render μπορεί να είναι μεγαλύτερος από αυτόν του προηγούμενου frame. Για να καλυφθεί το μεγάλο κόστος του rendering του επόμενου frame, πρέπει να αρχίσει να γίνεται render νωρίς για να μην χάσει το Vsync. Χρησιμοποιήθηκε το T_{motion} για το σκοπό αυτό στην Εξίσωση 5. Ωστόσο, όπως το frame rate είναι τόσο υψηλό όσο 90 Hz, η απόλυτη απόσταση που μπορεί να κινηθεί ο παίκτης κατά μήκος μιας κατεύθυνσης σε ένα χρόνο καρέ (δηλαδή, 1/90 δευτερόλεπτα) είναι αρκετά μικρή και επομένως έχει περιορισμένο αντίκτυπο στις αλλαγές περιεχομένου του επόμενου frame. Όμως, η περιστροφή του HMD μπορεί να επηρεάσει σημαντικά το περιεχόμενο του επόμενου καρέ λόγω της αλλαγής της οπτικής γωνίας. Επομένως στο σχέδιο αυτό εξετάστηκαν μόνο οι αλλαγές της οπτικής γωνίας. Όπως φαίνεται στην εξίσωση 6, το T_{motion} προσδιορίζεται από την αλλαγή οπτική γωνίας $\Delta\theta^{n}$ μαζί με μια σταθερή παράμετρο κλίμακας k.

Τέλος, χρησιμοποιήθηκε μια παράμετρο κλίμακας cc σαν low pass φίλτρο υπολογίζοντας την τιμή T_{shift}, όπως φαίνεται στην Εξίσωση 4.

2.4.2 Coterie

Το coterie design είναι ένα framework το οποίο πετυχαίνει υψηλής ποιότητας multiplayer VR για κινητές συσκευές.

Η υποστήριξη παιχνιδιών VR υψηλής ποιότητας σε κινητές συσκευές είναι απαιτητικό έργο ακόμη και για έναν παίκτη λόγω του περιορισμένου mobile hardware και τις περιορισμένες ασύρματες τεχνολογίες.

Η απλούστερη προσέγγιση για την υποστήριξη της εικονικής πραγματικότητας σε κινητές συσκευές είναι η εκτέλεση ολοκλήρου του rendering στην κινητή συσκευή, το οποίο έχει γίνει διαθέσιμο σε συστήματα VR όπως το Google Daydream και το Samsung Gear VR. Παρόλα αυτά λόγω των περιορισμένων δυνατοτήτων CPU/GPU κινητών συσκευών, τέτοια συστήματα μπορούν να υποστηρίξουν μόνο εφαρμογές VR χαμηλής ανάλυσης.

Ειδικότερα η προφανής εναλλακτική προσέγγιση είναι να αποφευχθεί η υπερφόρτωση της κινητής συσκευής CPU/GPU συνολικά και αντ' αυτού να ξεφορτώσει τον έντονο υπολογιστικό φόρτο του rendering σε έναν ισχυρό server και να μεταδώσει ασύρματα τα rendered frames στο headset ή την κινητή συσκευή που χρησιμοποιείται ως συσκευή προβολής (display device). Ωστόσο η υποστήριξη εφαρμογών VR υψηλής ανάλυσης με αυτόν τον τρόπο θα απαιτούσαν bandwidth πολλών Gbps για να ικανοποιηθεί ο περιορισμός της latency για κάθε frame. Σε αυτήν την περίπτωση ο τόσο υψηλός ρυθμός δεδομένων θα εξαντλούσε την CPU της κινητής συσκευής από την επεξεργασία πακέτων, π.χ., Furion εκτιμά ότι η επεξεργασία 4 Gbps θα απαιτούσε 16 ισοδύναμους πυρήνες που λειτουργούν με 70% χρήση σε ένα τηλέφωνο Pixel.

Αναλυτικότερα το Furion είναι η πρώτη σχεδίαση συστήματος VR που πληροί και τις τρεις απαιτήσεις QoE σε κινητές συσκευές και WiFi, για εφαρμογές VR με ανάλυση 4K για έναν παίκτη.

Για να ξεπεραστεί η εξάντληση των πόρων των local rendering και remote rendering, το Furion χρησιμοποιεί μια split αρχιτεκτονική που χωρίζει το rendering μεταξύ της κινητής συσκευής και του server, με βάση κάποιες βασικές παρατηρήσεις (1) για τις περισσότερες εφαρμογές VR, το περιεχόμενο VR που αποδίδεται μπορεί να χωριστεί σε foreground interactions (FI) και background virtual environment (BE). (2) Τα FI προκαλούνται από τους παίκτες που χειρίζονται το controller ή σήματα από άλλους παίκτες και είναι τυχαία και δύσκολο να προβλεφθούν, ενώ το BE ενημερώνονται σύμφωνα με την κίνηση του χρήστη η οποία αλλάζει συνεχώς και είναι προβλέψιμη. (3) Τα FI είναι πολύ πιο ελαφριά στο rendering σε σύγκριση με τα BE.

Επιπλέον για την ενεργοποιήσει του pre-rendering και του prefetching των frames, μερικά VR συστήματα διακριτοποιούν τον εικονικό κόσμο στο VR παιχνίδι σε έναν πεπερασμένο αριθμό σημείων κανάβου (grid points), έτσι ώστε ο server απλώς χρειάζεται να κάνει pre-render τα frames όταν προβάλλονται από αυτά τα grid points. Με τη διακριτοποίηση εικονικού κόσμου, το Furion καταφέρνει 4K ανάλυση για VR σε φορητές συσκευές :

1. διαίρεση του φόρτου εργασίας του VR rendering όταν ένας παίκτης μετακινείται από το σημείο του πλέγματος i στο επόμενο σημείο του πλέγματος i+1 σε FI και BE,

2. αξιοποίηση την GPU της κινητής συσκευής για απόδοση FI στο grid point i,

3. αξιοποίηση της remote rendering μηχανής στον server για pre-render και prefetch του BE για το grid point i+1

4. αποκωδικοποίηση των προανακτημένων BE για το grid point i στην κινητή συσκευή

5. συνδυασμός FI και BE για το grid point i στην κινητή συσκευή για να δημιουργηθεί το τελικό frame.

Επιπλέον, μετά την άφιξη του παίχτη στο επόμενο grid point, ο παίχτης μπορεί να αλλάξει τον προσανατολισμό του κεφαλιού του, κάτι που είναι δύσκολο να προβλεφθεί, Το Furion εκτελεί prefetch σε πανοραμικά frame 4K (3840×2160 pixel) του BE από τον server τα οποία μπορούν να περικοπούν για να αποδώσουν frame από οποιοδήποτε Field-of-View (FoV) για τα BE σχεδόν χωρίς κόστος ή καθυστέρηση.

2.4.2.1 Προκλήσεις υποστήριξης εικονικής πραγματικότητας για πολλούς παίκτες

Ο πιο προφανής τρόπος για επέκταση μια εφαρμογής VR για έναν παίκτη όπως το Furion για την υποστήριξη εφαρμογών VR για πολλούς παίκτες είναι η επανάληψη N-φορές της αρχιτεκτονικής client-server, ένα για το καθένα παίχτη

Επιπλέον το Multi-Furion, επεκτείνει το single-player Furion για υποστήριξη πολλών παικτών προσθέτοντας υποστήριξη για παίκτες (κινητές συσκευές) ανταλλαγής FI έτσι ώστε κάθε παίκτης μπορεί να αποδώσει ανεξάρτητα το FI όλων των παικτών τοπικά, το οποίο στη συνέχεια θα ενσωματωθεί με τα prefetched BE όπως πριν.

Σημαντικό αναφοράς είναι ότι εφαρμόστηκε ανταλλαγή FI μεταξύ των κινητών συσκευών Multi-Furion μέσω του framework photon unity networking (PUN). Ειδικότερα το PUN ενεργοποιεί το συγχρονισμό αντικειμένων στην Unity με χαμηλή latency. Χρησιμοποιώντας το PUN, ένα αντικείμενο FI μπορεί να συγχρονίσει τη θέση, την περιστροφή και το animation με τα απομακρυσμένα αντίγραφα του για κάθε frame.

Στην συνέχεια θα δούμε ένα πείραμα που πραγματοποιήθηκε για να καταλάβουμε πόσο καλά προσεγγίζει το MultiFurion , κινητά και οι Thin-client προσεγγίζουν σε κλίμακα την υποστήριξη εφαρμογών VR πολλαπλών παικτών, μεταφέρθηκαν τρεις δημοφιλείς εφαρμογές VR υψηλής ανάλυσης του Unity Store, Viking Village, CTS Procedural World και Racing Mountain, χρησιμοποιώντας το Google Daydream SDK.

Και για τις τρεις εκδόσεις κάθε εφαρμογής, ενσωματώθηκε το PUN κατεβασμένο από την Unity με foreground interaction για την υποστήριξη πολλών παικτών. Για τις εκδόσεις Thin-client και Multi-Furion, ο server χρησιμοποιεί το H.264 για κωδικοποίηση rendered frame πριν μεταδοθούν στον client.

Τα υλικά χαρακτηριστικά που περιλαμβάνονται σε αυτό το πείραμα είναι 4 Pixel 2 τηλέφωνα και έναν επιτραπέζιο υπολογιστή ως server με Intel 6-core Xeon CPU E5-1650, κάρτα γραφικών NVIDIA GeForce GTX 1080 Ti, και 16 GB RAM που τρέχει Windows 10 και Unity 5.6. Χρησιμοποιώντας iperf, μετρήθηκαν τα TCP δεδομένα που μεταφέρθηκαν ανά δευτερόλεπτο (throughput) από το server με 802.11ac (Wi-Fi) να είναι περίπου 500 Mbps.

Κάθε παιχνίδι παίζεται με τον ίδιο τρόπο 5 φορές για κάθε έκδοση, η καθεμία διαρκεί 10 λεπτά. Ο Πίνακας στην εικόνα 6 δείχνει την μέση επίδοση των τριών μεθόδων VR για την υποστήριξη των τριών εφαρμογών VR υψηλής ανάλυσης για 1 και 2 παίκτες. Κάνουμε τις παρακάτω παρατηρήσεις.

(1) Η απόδοση της Mobile έκδοσης παραμένει περίπου η ίδια όταν τρέχει με 1 παίκτη και με 2 παίκτες όσον αφορά τα FPS την καθυστέρηση μεταξύ των καρέ (inter-frame latency) και τον υπολογιστικό φόρτο CPU/GPU.

(2) Ως αναμενόμενο, η έκδοση Thin-client παρουσιάζει αύξηση σχεδόν 2 φορές σε διαδικτυακή καθυστέρηση μεταφοράς (network transfer latency) σε κάθε frame, επειδή ο server πρέπει να μεταφέρει 2 frame (σε 2 παίκτες) σε κάθε rendering interval, το οποίο διογκώνει το inter-frame latency από το εύρος 41–50ms έως 52–64ms, μειώνοντας τα FPS από 20 περίπου κάτω στα 16 περίπου.

(3) Όπως το Thin-client, το Multi-Furion επίσης υποφέρει σχεδόν 2 φορές αύξηση στην network transfer latency στο prefetched BE από τον server, λόγω του διπλασιασμένου φορτίου στο δίκτυο, το οποίο διογκώνει την inter-frame latency από 16,7ms για 60 FPS έως σχεδόν 21 ms, που μειώνει το FPS σε 42–48.

(4) Σε όλα τα πειράματα, το φορτίο του CPU του server παραμένει κάτω από 12% και ως εκ τούτου δεν είναι η αιτία καθυστέρησης του δικτύου. Η GPU του server δεν δημιουργεί bottleneck στο Multi-Furion που ανακτά pre-rendered BE frame.

(5) Στο Multi-Furion, η GPU του κινητού χρησιμοποιείται ελαφρά, γύρω στο 15%.

App (players)	FPS	InterFrame lat. (ms)	Phone CPU load (%)	Phone GPU load (%)	Per Frame size (KB)	Net. delay (ms)
Mobile						
Viking (1P)	26,0	38,2	17,3	88,0	null	null
CTS (1P)	24,0	42,0	9,5	99,0	null	null
Racing (1P)	27,0	38,2	10,3	92,0	null	null
Viking (2P)	24,0	42,5	19,6	88,0	null	null
CTS (2P)	21,0	48,3	9,6	99,0	null	null
Racing (2P)	25,0	40,3	13,2	93,0	null	null
Thin-client						
Viking (1P)	24,0	41,1	25,1	7,0	586,0	9,7
CTS (1P)	20,0	50,3	24,5	9,5	590,0	9,9
Racing (1P)	20,0	50,0	21,2	10,9	680,0	11,3
Viking (2P)	19,0	52,2	25,0	8,0	586,0	19,8
CTS (2P)	16,0	59,0	30,4	9,7	590,0	20,1
Racing (2P)	15,0	64,1	21,2	13,9	680,0	21,2
Multi-Furion						
Viking (1P)	60,0	16,0	23,2	13,2	550	9,2
CTS (1P)	60,0	16,6	29,7	12,9	440	7,5
Racing (1P)	60,0	16,5	29,4	14,0	564	9,3
Viking (2P)	45,0	22,2	31,0	16,4	550	18,3
CTS (2P)	48,0	20,8	32,9	15,4	440	16,2
Racing (2P)	42,0	23,8	32,2	14,8	564	18,5

Πίνακας 4: Επίδοση Mobile, thin client και Multi-Furion σε Pixel 2 κινητό

(Πηγή:https://www.researchgate.net/publication/339921061_Coterie_Exploiting_Frame_Similarity_to_Enable_High-Quality_Multiplayer_VR_on_Commodity_Mobile_Devices Meng et al [2020])

2.4.2.2 Exploiting frame similarity

Σε αυτό το σημείο αναγνωρίζουμε πως οι μετρήσεις δείχνουν ότι το κλειδί για την υποστήριξη εικονικής πραγματικότητας για πολλούς παίκτες είναι η μείωση των απαιτήσεων του bandwidth του δικτύου, το οποίο κλιμακώνεται με τον αριθμό των παικτών, το μέγεθος του κάθε frame (per-frame size) και πόσο συχνά ένα frame μεταφέρεται από τον server σε κάθε client. Αφού τα πλαίσια είναι ήδη συμπιεσμένα με την καλύτερη διαθέσιμη τεχνική συμπίεσης βίντεο, πρέπει να μειωθεί η συχνότητα μεταφοράς των frames.

Αν τα BE frame για κοντινά grid points στον εικονικό κόσμο του παιχνιδιού είναι αρκετά παρόμοια, ο client μπορεί να επαναχρησιμοποιηθεί τα prefetched BE frame για ένα grid point στο επόμενο γειτονικό σημείο για να μειωθεί η συχνότητα των prefetched BE frames. Συνοψίζοντας η αποτελεσματικότητα μιας τέτοιας προσέγγισης βασίζεται στην απάντηση των δύο παρακάτω ερωτήσεων:

1. Πόσο παρόμοια είναι τα BE frames για κοντινές τοποθεσίες;

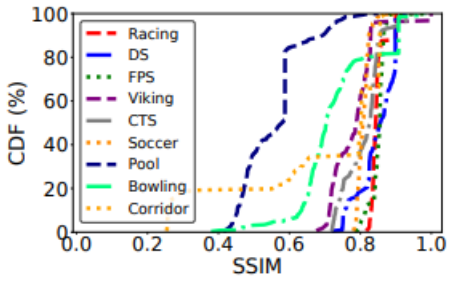
2. Πώς να εκθέσουμε στο μέγιστο την ομοιότητα μεταξύ των BE frame για να μειωθεί ο φόρτος δικτύου σε εφαρμογές VR για πολλούς παίκτες;

Βαθμός ομοιότητας των BE frames για κοντινές τοποθεσίες

Αρχικά μετρήθηκε η ομοιότητα μεταξύ των BE frame για γειτονικά grid points που διανύονται από έναν παίχτη, για 6 εφαρμογές VR σε εξωτερικούς και 3 σε εσωτερικούς χώρους από Unity Asset Store. Ο παίκτης παίζει κάθε παιχνίδι με ανάλυση 4K σε τηλέφωνο Pixel 2 για 10 λεπτά. Στην συνέχεια καταγράφτηκε η πορεία του παίκτη στον εικονικό κόσμο κατά τη διάρκεια του παιχνιδιού χρησιμοποιώντας το Multi-Furion. Έπειτα δημιουργήθηκε το πανοραμικό BE frame εκτός σύνδεσης για κάθε grid point στην πορεία του παίκτη στον server και υπολογίστηκε η ομοιότητα μεταξύ των γειτονικών BE frames.

Ειδικότερα για την μέτρηση της ομοιότητας μεταξύ ενός ζεύγους frames επιλέχθηκε το structural similarity index measure (SSIM). Το SSIM είναι μια de facto μέτρηση η οποία χρησιμοποιείται για την μέτρηση της ομοιότητας μεταξύ δύο εικόνων με μοντελοποίηση της αντίληψης του ανθρώπινου ματιού. Μια τιμή SSIM μεγαλύτερη από 0,90 υποδηλώνει ότι το παραμορφωμένο frame προσεγγίζει το αρχικό frame υψηλής ποιότητας και παρέχει "καλή" οπτική ποιότητα.

Συμπληρωματικά το σχήμα στην εικόνα 6 απεικονίζει το CDF των ομοιοτήτων μεταξύ του καθενός BE frame και του επόμενου διπλανού του frame στην πορεία. Παρατηρούμε σχεδόν μηδενική ομοιότητα, δηλαδή. Το ποσοστό των BE frame που εμφανίζουν μεγαλύτερη SSIM τιμή από 0,90 σε σύγκριση με το διπλανό του BE frame στην πορεία του παίκτη κυμαίνεται μεταξύ 0% και 20% για τα 9 παιχνίδια VR.



Εικόνα 6: Ομοιότητες παρακείμενων background frame

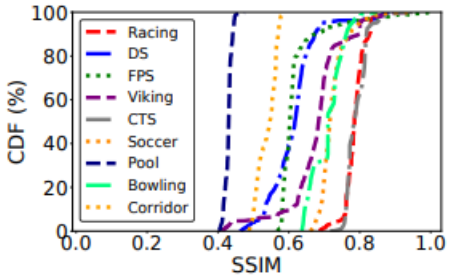
(Πηγή:https://www.researchgate.net/publication/339921061_Coterie_Exploiting_Frame_Similarity_to_Enable_High-Quality_Multiplayer_VR_on_Commodity_Mobile_Devices Meng et al [2020])

Για μια κατηγορία υπαίθριων χώρων multiplayer VR εφαρμογών, παρατηρούμε ότι οι πολλαπλοί παίκτες συνήθως αλληλοεπιδρούν στενά και ως εκ τούτου τείνουν να βρίσκονται σε στενή γειτνίαση μεταξύ τους στον εικονικό κόσμο ανά πάσα στιγμή κατά τη διάρκεια του παιχνιδιού. Για παράδειγμα, σε ένα τυπικό αυτοκίνητο αγωνιστικό παιχνίδι, πολλά αυτοκίνητα θα κυνηγήσουν το ένα το άλλο και σε ένα παιχνίδι περιπέτειας, πολλά avatar ακολουθούν ο ένας τον άλλον για να επιβιώσουν και να νικήσουν τους εχθρούς τους.

Για τέτοια παιχνίδια για πολλούς παίκτες, ένα frame που γίνεται rendered για έναν παίκτη μπορεί ενδεχομένως να είναι αρκετά παρόμοιο με αυτό για ένα κοντινό παίκτη και ως εκ τούτου επαναχρησιμοποιείται από τον άλλο παίκτη. Με κίνητρο αυτή την παρατήρηση, μετρήθηκε στη συνέχεια η ομοιότητα μεταξύ των BE frame για δύο

παίκτες που παίζουν τις ίδιες 9 εφαρμογές VR, όπου δύο παίκτες παίζουν κάθε παιχνίδι σε Pixel 2 κινητά για 10 λεπτά, ταυτόχρονα.

Το σχήμα στην εικόνα 7 απεικονίζει το CDF των ομοιοτήτων στην καλύτερη περίπτωση μεταξύ των δύο παικτών. Η ομοιότητα των frame μεταξύ των παικτών είναι σχεδόν μηδενική, δηλαδή το ποσοστό των BE frame του παίκτη 1 που έχουν τιμές SSIM μεγαλύτερη από 0,90 είναι κοντά στο 0% για τα 9 παιχνίδια VR.



Εικόνα 7: Καλύτερη περίπτωση ομοιοτήτων των background frame

(Πηγή:https://www.researchgate.net/publication/339921061_Coterie_Exploiting_Frame_Similarity_to_Enable_High-Quality_Multiplayer_VR_on_Commodity_Mobile_Devices Meng et al [2020])

The "near object" effect

Πολλά BE frames περιέχουν αντικείμενα (Unity assets) κοντά στον παίκτη στον εικονικό κόσμο, εξαιτίας των οποίων ακόμη και μια μικρή μετατόπιση της θέσης του παίκτη μπορεί να οδηγήσει σε ορατή αλλαγή μεταξύ των frame. Το παραπάνω φαινόμενο "κοντινού αντικειμένου" μπορεί να εξηγηθεί από τεχνικές που χρησιμοποιούνται από μηχανές VR όπως η τεχνική που χρησιμοποιεί η Unity στο rendering των frames, που ονομάζεται προοπτική προβολή (perspective projection). Η τεχνική μιμείται τον τρόπο με τον οποίο βλέπουν τα ανθρώπινα μάτια τον κόσμο απεικονίζοντας ένα 3D grid point του κόσμου του παιχνιδιού σε ένα σημείο στο 2D frame. Συγκεκριμένα, μετατρέπει μακρινά αντικείμενα να φαίνονται μικρότερα και τα κοντινά αντικείμενα να φαίνονται μεγαλύτερα για να παρέχουν ρεαλισμό στα ανθρώπινα μάτια. Ως αποτέλεσμα, μια μικρή μετατόπιση ενός κοντινού αντικείμενου είναι πιο έντονη, δηλαδή αλλάζει πολύ περισσότερα pixel στο rendered frame, από αυτό των μακρινών αντικειμένων.

Το φαινόμενο του "κοντινού αντικειμένου" υποδηλώνει ότι αν διαχωριστεί το τμήμα του BE που είναι κοντά στη θέση του παίκτη από εκείνα που είναι μακριά, σε near BE και far BE, αντίστοιχα, η ομοιότητα των far BE frames που γίνεται rendered από κοντινές τοποθεσίες αυξάνεται σημαντικά. Ο διαχωρισμός σε near BE και far BE βασίζεται σε μια ακτίνα αποκοπής, όπου τα αντικείμενα εντός και εκτός της ακτίνας ανήκουν σε near BE και far BE, αντίστοιχα.

Για να γίνει ενός τέτοιος διαχωρισμός, πρέπει να αποφασιστεί πού θα πραγματοποιείται το rendering των near και far BE frames. Εφόσον το far BE τείνει να περιέχει πολύ περισσότερα αντικείμενα από το near BE, ειδικά για εφαρμογές VR εξωτερικού χώρου. Το rendering των far BE θα παραμείνει υπολογιστικά απαγορευτικό για την κινητή συσκευή και ως εκ τούτου θα πρέπει να εκτελείται στο server

Αντίθετα το near BE περιέχει λιγότερα αντικείμενα από το far BE και επομένως μπορεί να γίνει rendered είτε στην κινητή συσκευή είτε στον server. Ωστόσο, επειδή τα near BE είναι κοντά στο σημείο προβολής, ένα near BE frame θα καταλάβει σημαντικό μέρος του τελικού frame. Ως αποτέλεσμα τα near BE θα παραμείνουν μεγάλα. Συμπερασματικά αποτελέσματά τους δείχνουν ότι τα near και far BE frame είναι συγκρίσιμα μεγέθη, πιο συγκεκριμένα περίπου το μισό του αρχικού BE frame. Συνεπώς το rendering των near BE στον server δεν θα μειώσει σε μεγάλο βαθμό τον διαδικτυακό φόρτο και το rendering των near BE θα πρέπει να συμβεί στην κινητή συσκευή. Η εναπομείνασα πρόκληση σχεδιασμού είναι ο τρόπος προσδιορισμού της ακτίνας αποκοπής που χωρίζει τα κοντινά αντικείμενα στον εικονικό κόσμο από τα μακρινά.

Διαισθητικά, όσο μεγαλύτερη η ακτίνα αποκοπής, τόσο πιο μακριά είναι τα αντικείμενα στο far BE, επιπλέον τόσο πιο παρόμοια θα είναι τα far BE όταν τα βλέπει κανείς από κοντινές τοποθεσίες. Επιλέχθηκαν τέσσερις τυχαίες τοποθεσίες στο Viking village. Η τιμή SSIM μεταξύ δύο γειτονικών far BE frame στα τέσσερα επιλεγμένα σημεία αυξάνεται γρήγορα και μονότονα με την ακτίνα αποκοπής, από 0,68, 0,63, 0,75 και 0,83 στην ακτίνα αποκοπής 0 (δηλαδή ολόκληρο το BE είναι το far BE) έως πάνω από 0,9 στην ακτίνα αποκοπής των 4 μέτρων. Όμως, μια μεγαλύτερη ακτίνα θα έχει ως αποτέλεσμα υψηλότερο φόρτο στο rendering στην κινητή συσκευή αφού το near BE θα περιέχει περισσότερα αντικείμενα. Ο συμβιβασμός αυτός υποδηλώνει ότι πρέπει να χρησιμοποιηθεί η μεγαλύτερη πιθανή ακτίνα αποκοπής που δεν προκαλεί το χρόνο απόδοσης RT (rendering time) στην κινητή συσκευή να υπερβεί τον περιορισμό της latency, δηλαδή 16,7 ms:

$$\mathrm{RT_{FI}}+\mathrm{RT_{NearBE}}<16.7\mathrm{ms}\ (7)$$

Ο παραπάνω περιορισμός στην ακτίνα αποκοπής παρουσιάζει ότι η σωστή επιλογή αποκοπής εξαρτάται από την εφαρμογή και τη συσκευή, όπως και το rendering time των FI και near BE. Αυτό με τη σειρά του δείχνει ότι η αποκοπή πρέπει να καθορίζεται για κάθε εφαρμογή εκτός σύνδεσης, δηλαδή κατά την εγκατάσταση της εφαρμογής, με βάση τις δυνατότητες της συσκευής.

Για κάθε εφαρμογή, το rendering time για Το FI καθορίζεται από τη φύση του FI που επιτρέπεται στο παιχνίδι και γενικά δεν εξαρτάται από την τοποθεσία στον εικονικό κόσμο. Σημαντικό αναφοράς είναι πως, μπορεί πρώτα να προσδιοριστεί πειραματικά το $\mathrm{RT_{FI}}$ σε μια δεδομένη συσκευή για την εύρεση ενός άνω ορίου. Για παράδειγμα, για τις τρεις εφαρμογές VR ανάλυσης 4K, μετρήθηκε το $\mathrm{RT_{FI}}$ του gameplay του παίκτη ώστε να είναι κάτω από 4 ms στο Pixel 2, το οποίο δίνει έναν χρονικό περιορισμό στο rendering του near BE στο Pixel 2 για τον προσδιορισμό της ακτίνα αποκοπής:

$$\mathrm{RT_{NearBE}}<16.7\mathrm{ms}-\mathrm{RT_{FI}}=16.7\mathrm{ms}-4\mathrm{ms}=12.7\mathrm{ms}\ (8)$$

Αρχικά, εφόσον το rendering speed συσχετίζεται με τον αριθμό τριγώνων των αντικειμένων, πρώτα μετριέται η πυκνότητα του αντικειμένου σε μια τοποθεσία στο παιχνίδι με τον αριθμό τριγώνων σε μια σταθερή ακτίνα αποκοπής. Παρατηρείται ότι η πυκνότητα των αντικειμένων στον εικονικό κόσμο των VR παιχνιδιών μπορεί να διαφέρει σημαντικά. Αυτό υποδηλώνει ότι η χρήση μιας ενιαίας ακτίνας αποκοπής που

ικανοποιεί τον παραπάνω περιορισμό για τον εικονικό κόσμο θα είναι αναποτελεσματική καθώς θα καταλήξουμε με μικρότερες ακτίνες από τις απαραίτητες για πολλές τοποθεσίες. Από την άλλη πλευρά, προσαρμόζοντας μία ακτίνα αποκοπής για κάθε τοποθεσία στον εικονικό κόσμο είναι υπολογιστικά ανέφικτο καθώς τα παιχνίδια VR μπορούν να περιέχουν έως και εκατοντάδες εκατομμύρια grid points στον εικονικό κόσμο.

Ιδιαίτερα σημαντική παρατήρηση είναι ότι αν και η πυκνότητα του αντικειμένου μπορεί να ποικίλλει σε διαφορετικές τοποθεσίες του κόσμου του παιχνιδιού, αυτή αλλάζει σταδιακά και τείνει να είναι ομοιόμορφη μέσα σε μια μικρή περιοχή. Με βάση αυτήν την παρατήρηση, προτείνεται ένα προσαρμοστικό σχέδιο αποκοπής που μειώνει δραστικά τον αριθμό των ακτινών αποκοπείς που πρέπει να υπολογιστούν. Ειδικότερα, το σχέδιο διαιρεί επαναλαμβανόμενα τον εικονικό κόσμο του παιχνιδιού μέχρι οι ακτίνες αποκοπείς για διαφορετικές τοποθεσίες με κάθε υποπεριοχή να γίνουν περίπου ομοιόμορφες.

Το προσαρμοστικό σχέδιο λειτουργεί ως εξής. Δεδομένου ότι οι παίκτες κινούνται σε 2D στον εικονικό κόσμο σε τυπικά παιχνίδια VR, η επαναλαμβανόμενη διαίρεση του εικονικού κόσμου θα είναι 2D. Η διαδικασία διαίρεσης είναι επαναλαμβανόμενη, αρχικά καλείται με ολόκληρο τον κόσμο του παιχνιδιού. Σε κάθε επίκληση κάνει δειγματοληψία K τυχαίες τοποθεσίες από την περιοχή του παιχνιδιού, υπολογίζει την μέγιστη ακτίνα αποκοπής για κάθε τοποθεσία που ικανοποιεί τον περιορισμό 1, και ελέγχει εάν οι ακτίνες για τις K τοποθεσίες είναι παρόμοιες. Στην περίπτωση που είναι, καταγράφει την ελάχιστη ακτίνα ως ακτίνα για την περιοχή και επιστρέφει. Σε αντίθετη περίπτωση, η πυκνότητα των αντικειμένων στην περιοχή είναι πιθανώς άνιση και η διαδικασία χωρίζει την περιοχή σε 4 υποπεριοχές ίσου μεγέθους και επαναλαμβάνεται σε αυτές. Αναλυτικότερα, οι διαμερισμένες υποπεριοχές σχηματίζουν ένα quadtree και οι υποπεριοχές που δεν μπορούν να χωριστούν περαιτέρω αναφέρονται ως "leaf region".

Συνοπτικά, το προσαρμοστικό σχέδιο αποκοπής δημιουργεί ακτίνες αποκοπής σύμφωνα με τη μεταβαλλόμενη πυκνότητα των αντικειμένων στον κόσμο του παιχνιδιού για να ελαχιστοποιηθεί ο συνολικός αριθμός ακτινών που πρέπει να υπολογιστούν για τον διαχωρισμό far/near BE frame σε διαφορετικές τοποθεσίες στον κόσμο του παιχνιδιού ενώ ταυτόχρονα μεγιστοποιεί την ακτίνα αποκοπής για κάθε leaf region και ως εκ τούτου τις τοποθεσίες εντός αυτών. Η μεγιστοποιημένη ακτίνα αποκοπής σε κάθε leaf region οδηγεί σε υψηλή ομοιότητα μεταξύ των far BE frame σε κοντινές τοποθεσίες και ως εκ τούτου αυξάνει το hit ratio στο frame cache .

Το προσαρμοστικό σχέδιο αποκοπής είναι εξαιρετικά αποτελεσματικό, για το μεγαλύτερο παιχνίδι VR που χρησιμοποιήθηκε στο πείραμα, το CTS μειώνει τον αριθμό των υπολογισμών αποκοπής που απαιτούνται από 268 εκατομμύρια τοποθεσίες στον εικονικό κόσμο σε 235 υποπεριοχές. Το σχέδιο καθιστά εφικτό τον υπολογισμό της ακτίνας αποκοπής εκτός σύνδεσης.

2.4.2.3 Αρχιτεκτονική του Coterie

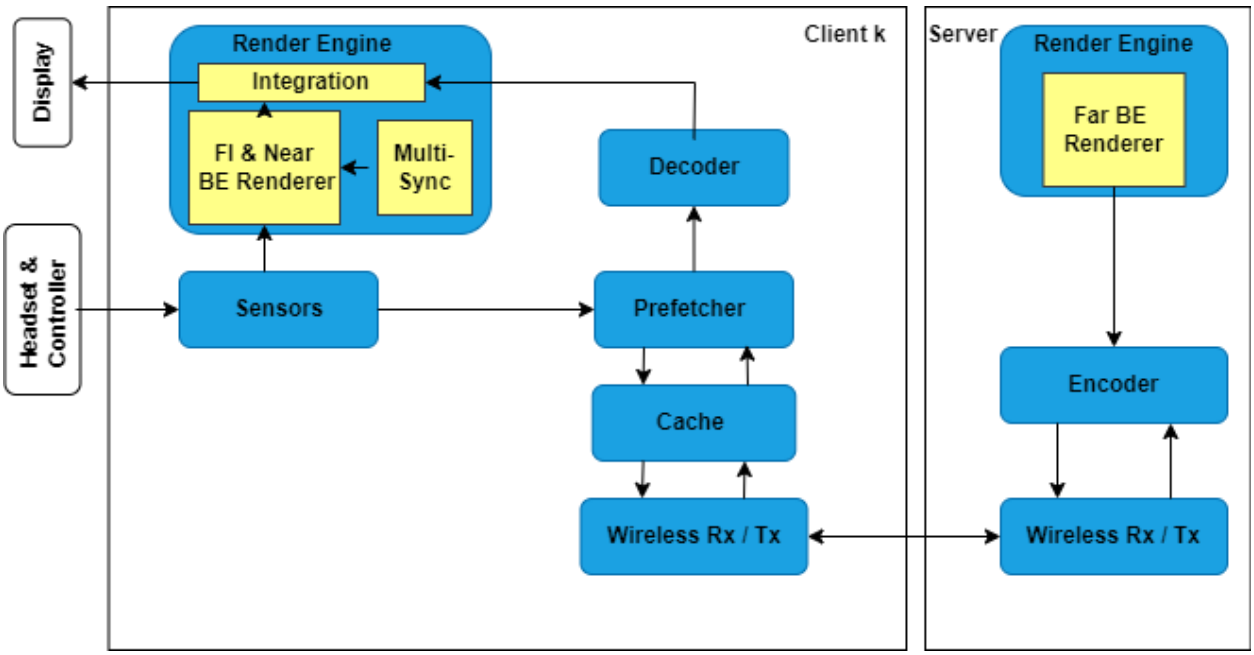
Ο στόχος του σχεδιασμού Coterie είναι να ενεργοποιήσει εφαρμογές VR για πολλούς παίκτες υψηλής ανάλυσης σε κινητά τηλέφωνα και WiFi δίκτυα μειώνοντας το απαιτούμενο bandwidth του δικτύου. Το Coterie επιτυγχάνει αυτόν τον στόχο μέσω μιας αρχιτεκτονικής rendering 3 επιπέδων, η οποία

(1) Διαχωρίζει το BE σε near BE και far BE για να εκθέσει την ομοιότητα των far BE frame

(2) render FI και near BE σε κινητές συσκευές

(3) αποθηκεύει προσωρινά και επαναχρησιμοποιεί παρόμοια far BE frame για να μειώσει σημαντικά το prefetching BE frame από τον server.

Η αρχιτεκτονική του συστήματος Coterie περιέχει έναν server ο οποίος τρέχει σε έναν επιτραπέζιο υπολογιστή και πολλαπλούς clients, όπου ο καθένας χρησιμοποιεί μια κινητή συσκευή.



Εικόνα 8: Αρχιτεκτονική Coterie

(Πηγή:https://www.researchgate.net/publication/339921061_Coterie_Exploiting_Frame_Similarity_to_Enable_High-Quality_Multiplayer_VR_on_Commodity_Mobile_Devices Meng et al [2020])

Coterie server.

Στην επεξεργασία εκτός σύνδεσης, ο server εκτελεί πρώτα τον προσαρμοστικό αλγόριθμο αποκοπής για την διαίρεση του εικονικού κόσμου σε πολλαπλές leaf regions και, στη συνέχεια, pre-rendered και προ-κωδικοποιεί πανοραμικά far BE frame για όλα τα grid points που μπορεί να φτάσει ο παίκτης. Κατά τη διάρκεια του παιχνιδιού, ο server ανταποκρίνεται σε αιτήματα για far BE frames από οποιονδήποτε client με κωδικοποιημένα prerendered πανοραμικά far BE frames χρησιμοποιώντας το πρωτόκολλο TCP.

Coterie Client.

Κατά τη διάρκεια κάθε χρονικού παραθύρου όταν ένας παίκτης μετακινείται από το ένα grid point στον εικονικό κόσμο στο επόμενο, ο client κάνει render το frame για το επόμενο grid point εκτελώντας ταυτόχρονα τέσσερις εργασίες, ακολουθούμενες από συγχώνευση πλαισίων:

1. FI και near BE rendering: Οι αισθητήρες συλλέγουν τη νέα πόζα από το headset και την κίνηση και τα FI από το χειριστήριο, που ενεργοποιούν την Render engine για να ξεκινήσει το rendering για FI και near BE για το επόμενο grid point.

2. Decoding: Το prefetched συμπιεσμένο far BE frame για το επόμενο grid point διαβάζεται από το frame cache και αποστέλλεται στο ανώτερο επίπεδο.

3. Prefetching και caching: Ο prefetcher καθορίζει τα far BE frames για τα γειτονικά σημεία του επόμενου grid point και αναζητά στο frame cache για το συγκεκριμένο ή κάποιο παρόμοιο frame. Εάν όλα τα απαραίτητα frame βρίσκονται στο frame cache, το prefetched παραλείπεται. Διαφορετικά, το αίτημα προανάκτησης αποστέλλεται στον server και τα frame αποθηκεύονται στην μνήμη cache.

4. Συγχρονισμός FI: Το FI των παικτών συγχρονίζεται μέσω PUN χρησιμοποιώντας το πρωτόκολλο UDP σε όλες τις κινητές συσκευές μέσω του server.

5. Συγχώνευση: Το αποκωδικοποιημένο far BE frame συγχωνεύεται με το τοπικά rendered FI και near BE στην rendering engine, η οποία στη συνέχεια το προβάλη για κάθε μάτι του παίκτη.

Οι τέσσερις χρονικά κρίσιμες εργασίες εκτελούνται παράλληλα. Έτσι το latency για το rendering ενός νέου frame είναι:

$$T_{split\_render} = \max(T_{phone\_render\_FI} + T_{phone\_render\_NearBE}, T_{phone\_decode\_farBE}, T_{phone\_prefetch\_next\_farBE}, T_{phone\_sync\_FI}) + T_{merge} \quad (9)$$

Κεφάλαιο 3: Εργαλεία

3.1 Unity

Η Unity είναι μια cross-platform game engine που αναπτύχθηκε από την Unity Technologies. Δίνει την δυνατότητα δημιουργίας τρισδιάστατων και δισδιάστατων παιχνιδιών καθώς και διαδραστικές προσομοιώσεις. Η μηχανή είναι γραμμένη στην γλώσσα προγραμματισμού C++ αλλά το framework χρησιμοποιεί την C#. Τέλος περιέχει modules animation, networking, μηχανή φυσικής προσομοίωσης, input, Εικονική πραγματικότητα (VR), επαυξημένη πραγματικότητα (AR).

3.1.1 Rendering pipeline

Με τον όρο rendering pipeline εννοούμε τον συνδυασμό πολλαπλών σταδίων ή βημάτων που απαιτούνται για τη δημιουργία μιας δισδιάστατης εικόνας με βάση, μία συγκεκριμένη γεωμετρική περιγραφή μιας τρισδιάστατης σκηνής και μιας προσανατολισμένης εικονικής κάμερας. Επιπρόσθετα, το pipeline του DirectX (πάνω στο οποίο η Unity3D δημιουργεί τα γραφικά στοιχεία για λειτουργικά συστήματα Windows , Linux και Mac) είναι μια σειρά ενεργειών, οι οποίες συμβαίνουν σε διαδοχικά βήματα ως εξής.

Πρώτα, ο input assembler διαβάζει δεδομένα (κορυφές και δείκτες) τρισδιάστατων αντικειμένων από τη μνήμη και τα προσεγγίζει χρησιμοποιώντας τριγωνικά meshes. Όσο αυξάνονται τα τρίγωνα, τόσο καλύτερη η προσέγγιση αλλά αυξάνεται και η υπολογιστική ισχύς. Κάθε αντικείμενο τοποθετείται στη συνέχεια σε ένα τοπικό σύστημα συντεταγμένων με δικό του προσανατολισμό και διαστάσεις. Τελικά, όλα τα αντικείμενα συγκεντρώνονται σε ένα καθολικό σύστημα συντεταγμένων εφαρμόζοντας γεωμετρικούς μετασχηματισμούς.

Το δεύτερο βήμα σχετίζεται με την κάμερα. Το view space είναι ένα σύστημα συντεταγμένων στο οποίο η εικονική κάμερα μεταφράζεται ως η προέλευση του χώρου του εικονικού κόσμου. Τότε είναι δυνατό να γίνει διάκριση μεταξύ μπροστινής και πίσω πλευράς των αντικειμένων. Στο πίσω μέρος οι πλευρές δεν φαίνονται από την κάμερα οπότε όλα τα πίσω πολύγωνα δεν συμβάλλουν στην τελική εικόνα.

Το βήμα φωτισμού είναι ένα βασικό βήμα για τη δημιουργία μιας ρεαλιστικής σκηνής. Οι πηγές φωτός είναι απλά αντικείμενα, καθορισμένα στον χώρο του εικονικού κόσμου, που είναι συνδυασμός χρώματος, έντασης, κατεύθυνσης, εστίασης και θέσης. Αυτό το βήμα περιλαμβάνει μια επανάληψη διαδικασίας απορρόφησης και ανάκλασης φωτός ανάλογα με διάφορες παραμέτρους (π.χ. ομαλότητα ,υλικό επιφάνειας και γωνία πρόσπτωσης). Στη συνέχεια, αποφασίζεται ποια αντικείμενα πρέπει να απορρίπτονται από τη σκηνή σύμφωνα με τον υπολογισμό της προβολής frustum[2] .

[2] Η προβολή frustum είναι η περιοχή του χώρου στον εικονικό κόσμο που μπορεί να εμφανιστεί στην οθόνη

Ένα αντικείμενο μέσα στο frustum διατηρείται μέσα τη σκηνή, ενώ το αντικείμενο που βρίσκεται εκτός περιοχής του frustum καταστρέφεται.

Τέλος, η σκηνή γίνεται rendered με βάση την προοπτική προβολή των τρισδιάστατων κορυφών σε ένα παράθυρο δισδιάστατης προβολής μέσα στο frustum. Οι συντεταγμένες των κορυφών μετασχηματίζονται για να τοποθετήσουν την 2D σκηνή σε ένα ορθογώνιο παράθυρο στην οθόνη, το οποίο καλείται το παράθυρο προβολής, με κλίμακα και μετάθεση. Τα αποτελέσματα αυτού του σταδίου είναι pixel. Το rasterization μετατρέπει αυτά τα pixel σε συντεταγμένες οθόνης που σχηματίζουν μια λίστα τριγώνων, τα οποία πρέπει να χρωματιστούν.

3.2 WebRTC

3.2.1 Περιεχόμενα του WebRTC

WebRTC είναι συντομογραφία για το Web Real-Time Communication, το οποίο είναι ταυτόχρονα ένα API και ένα Πρωτόκολλο. Το πρωτόκολλο WebRTC είναι ένα σύνολο κανόνων για αμφίδρομη ασφαλή επικοινωνία σε πραγματικό χρόνο μεταξύ δύο WebRTC agents. Το WebRTC API επιτρέπει στους προγραμματιστές να χρησιμοποιούν το πρωτόκολλο WebRTC. Το WebRTC API χρησιμοποιεί ως γλώσσα προγραμματισμού την JavaScript.

3.2.2 Το πρωτόκολλο WebRTC

Το πρωτόκολλο WebRTC είναι μια συλλογή από πολλές τεχνολογίες . Χωρίζεται σε τέσσερα βήματα:

1. Signaling
2. Connecting
3. Secure
4. Communication

Τα βήματα που αναφέρθηκαν είναι διαδοχικά, πράγμα που σημαίνει ότι το προηγούμενο βήμα πρέπει να είναι 100% επιτυχημένο για να ξεκινήσει το επόμενο βήμα. Αξιοσημείωτο είναι το γεγονός για το WebRTC, ότι κάθε βήμα στην πραγματικότητα αποτελείται από πολλά άλλα πρωτόκολλα! Για την κατασκευή του WebRTC συνδυάστηκαν πολλές υπάρχουσες τεχνολογίες, υπό αυτή την έννοια μπορούμε να σκεφτούμε ότι το WebRTC είναι περισσότερο ένας συνδυασμός και διαμόρφωση μιας καλά κατανοητής τεχνολογίας που χρονολογείται από τις αρχές της δεκαετίας του 2000 παρά ως μια ολοκαίνουργια αυτόνομη διεργασία.

Signaling

Κατά την έναρξη της διαδικασίας, ένα WebRTC agent δεν έχει ιδέα με ποιον χρήστη πρόκειται να επικοινωνήσει ή με τι μέσο και για ποιο λόγο πρόκειται να επικοινωνήσουν. Το βήμα signaling λύνει αυτό το ζήτημα, με άλλα λόγια χρησιμοποιείται για την εκκίνηση της κλήσης, επιτρέποντας δύο ανεξάρτητους WebRTC agents να ξεκινήσουν να επικοινωνούνε.

Πιο συγκεκριμένα, το signaling χρησιμοποιεί ένα υπάρχον πρωτόκολλο απλού κειμένου που ονομάζεται SDP (Session Description Protocol). Κάθε μήνυμα SDP αποτελείται από ζεύγη κλειδιών/τιμών και περιέχει μια λίστα των media. Το SDP που ανταλλάσσουν οι δύο WebRTC agent περιέχει λεπτομέρειες όπως:

- Τα IP και Ports στα οποία έχει πρόσβαση ο agent (candidates).
- Ο αριθμός των track ήχου και βίντεο που επιθυμεί να στείλει ο agent.
- Οι κωδικοποιητές ήχου και βίντεο που υποστηρίζει κάθε πράκτορας.
- Οι τιμές που χρησιμοποιούνται κατά τη σύνδεση (uFrag/uPwd).
- Οι τιμές που χρησιμοποιούνται κατά την ασφάλιση (certificate fingerprint).

Είναι σημαντικό να σημειωθεί ότι, το signaling συνήθως συμβαίνει "εκτός ζώνης", το οποίο σημαίνει ότι οι εφαρμογές γενικά δεν χρησιμοποιούν το ίδιο το WebRTC για την ανταλλαγή signaling μηνυμάτων. Οποιαδήποτε αρχιτεκτονική κατάλληλη για την αποστολή μηνυμάτων μπορεί να προωθήσει τα SDP μεταξύ των συνδεόμενων peer, και πολλές εφαρμογές θα χρησιμοποιήσουν απλώς τις υπάρχουσες υποδομές τους για τη διευκόλυνση των συναλλαγών SDP μεταξύ των κατάλληλων client.

Connecting

Μόλις δύο WebRTC agent ανταλλάξουν SDP, έχουν αρκετές πληροφορίες για να προσπαθήσουν να συνδεθούν μεταξύ τους. Για να πραγματοποιηθεί αυτή η σύνδεση, το WebRTC χρησιμοποιεί μια άλλη καθιερωμένη τεχνολογία που ονομάζεται ICE (Interactive Connectivity Establishment).

Το ICE είναι ένα πρωτόκολλο που χρονολογείται προγενέστερα του WebRTC και επιτρέπει τη δημιουργία μιας απευθείας σύνδεσης μεταξύ δύο agent χωρίς κεντρικό server. Αυτοί οι δύο agent μπορεί να είναι στο ίδιο δίκτυο ή στην άλλη άκρη του κόσμου.

Όταν οι δύο agent έχουν δημιουργήσει με επιτυχία μια σύνδεση ICE, το WebRTC προχωρά στο επόμενο βήμα. (δημιουργία κρυπτογραφημένης μεταφοράς για κοινή χρήση ήχου, βίντεο και δεδομένα μεταξύ τους).

Securing

Ακολούθως, όταν επιτευχθεί η αμφίδρομη επικοινωνία (μέσω ICE), πρέπει να κάνουμε την επικοινωνία ασφαλή. Ζητούμενο που επιτυγχάνεται μέσω δύο ακόμη πρωτοκόλλων που επίσης έχουν προγενέστερη ημερομηνία δημιουργίας από το WebRTC. DTLS (Datagram Transport Layer Security) και SRTP (Secure Real-Time

Transport Protocol). Το πρώτο πρωτόκολλο, DTLS, είναι απλά TLS μέσω UDP (το TLS είναι το κρυπτογραφικό πρωτόκολλο που χρησιμοποιείται για την ασφάλεια της επικοινωνίας μέσω HTTPS). Το δεύτερο πρωτόκολλο, το SRTP, χρησιμοποιείται για τη διασφάλιση της κρυπτογράφησης του RTP (Real-time Transport Protocol) data packets.

Αρχικά, το WebRTC συνδέεται κάνοντας ένα handshake DTLS στη σύνδεση που ιδρύθηκε από το ICE. Έπειτα, το WebRTC βεβαιώνει ότι το πιστοποιητικό που ανταλλάσσεται μέσω DTLS ταιριάζει το κοινόχρηστο fingerprint μέσω signaling Αυτή η σύνδεση DTLS χρησιμοποιείται στη συνέχεια για μηνύματα DataChannel.

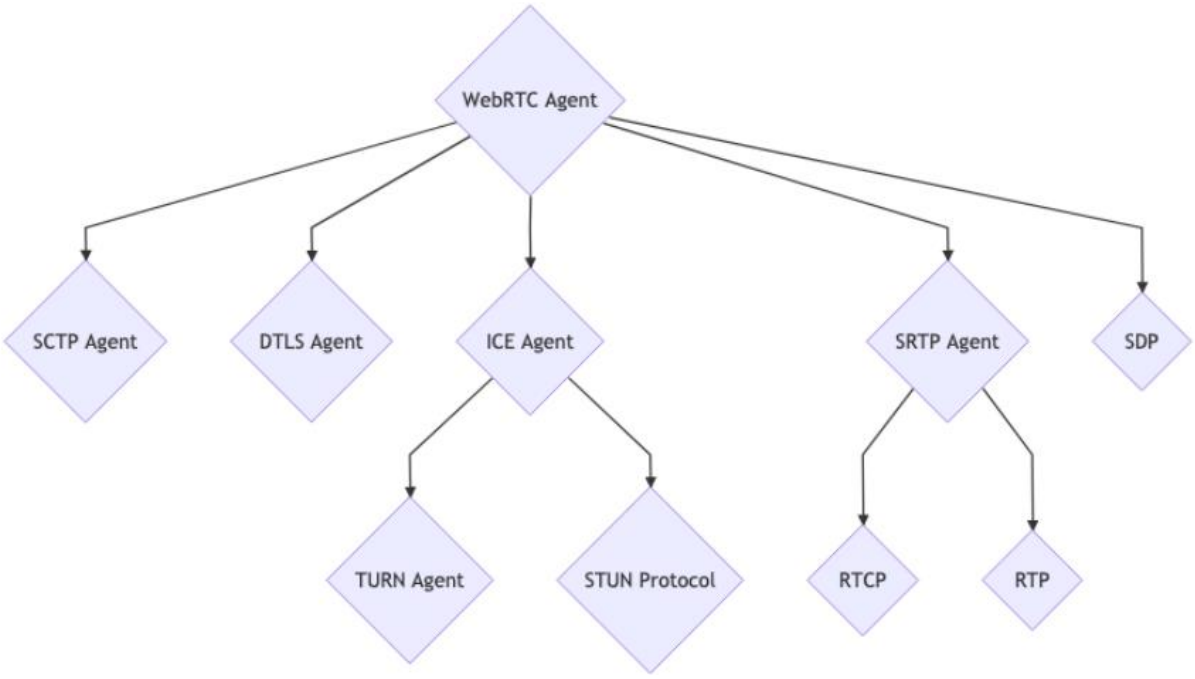
Κατόπιν, το WebRTC χρησιμοποιεί το πρωτόκολλο RTP, ασφαλισμένο με SRTP, για μετάδοση ήχου και βίντεο.

Communicating

Στο συγκεκριμένο σημείο έχουμε δύο WebRTC agent συνδεδεμένους και ασφαλείς, ξεκινάει η επικοινωνία. Πάλι, το WebRTC χρησιμοποιεί δύο προϋπάρχοντα πρωτόκολλα: RTP και SCTP (Stream Control Transmission Protocol). Χρησιμοποιείτε το RTP για την ανταλλαγή πολυμέσων κρυπτογραφημένα με SRTP, παράλληλα χρησιμοποιούμε το SCTP για αποστολή και λήψη μηνυμάτων DataChannel κρυπτογραφημένα με DTLS.

Το RTP είναι ένα αρκετά μικρό πρωτόκολλο, αλλά παρέχει τα απαραίτητα εργαλεία για εφαρμογή streaming σε πραγματικό χρόνο. Το πιο σημαντικό πράγμα για το RTP είναι ότι δίνει ευελιξία για τον προγραμματιστή, επιτρέποντάς του να χειριστεί την latency και την απώλεια πακέτων.

Το τελικό πρωτόκολλο στη στοίβα είναι SCTP. Το σημαντικό για το SCTP είναι ότι επιτρέπει την αναξιόπιστη, χωρίς σειρά παράδοση μηνυμάτων. Αυτό επιτρέπει στους προγραμματιστές να εξασφαλίσουν την απαραίτητη latency σε συστήματα Real-time.



Εικόνα 9: WebRTC Agent

(Πηγή: https://webrtcforthecurious.com/)

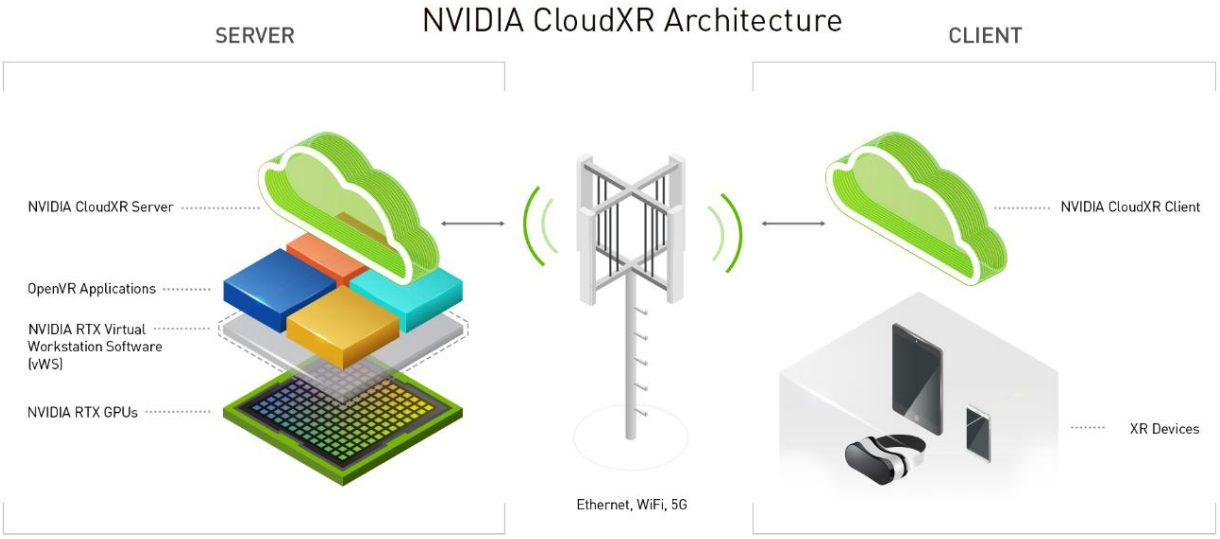
3.3 ALVR

Το ALVR[3] ή Air Light VR είναι ένα open source λογισμικό το οποίο επιτρέπει να τρέξουν εφαρμογές VR του υπολογιστή σε ασύρματο VR headset. Στην ουσία αξιοποιεί την υπολογιστική ισχύ του υπολογιστή καθώς το ασύρματο VR δεν έχει αρκετή από μόνο του και δίνει την δυνατότητα να μεταδώσει περιεχόμενο VR από τον υπολογιστή στο headset εξαλείφοντας την ανάγκη για σύνδεση μέσω καλωδίου.

Το ALVR περιέχει δύο εφαρμογές, μία για τον server και μία για τον client. Η εφαρμογή του server πρέπει να εγκατασταθεί στον υπολογιστή που πραγματοποιείται το rendering των γραφικών και είναι υπεύθυνη για την επικοινωνία με το VR headset καθώς και για το rendering του VR περιεχομένου. Η εφαρμογή του client πρέπει να εγκατασταθεί στο VR headset και είναι υπεύθυνη για να δεχθεί το VR περιεχόμενο που μεταδίδεται από τον υπολογιστή και να το εμφάνιση στο headset. Για να πραγματοποιηθεί η σύνδεση πρέπει ο υπολογιστής και το headset να είναι συνδεδεμένα στο ίδιο Wi-Fi δίκτυο. Όταν ρυθμιστούν ο server και ο client και συνδεθούν μεταξύ τους τότε ο ALVR server ξεκινάει να μεταδίδει το περιεχόμενο που αποδίδεται από τον server στον client μέσω του Wi-Fi δικτύου. Στη συνέχεια ο ALVR client λαμβάνει το περιεχόμενο που μεταδίδεται και το εμφανίζει στο headset σε περιβάλλον VR. Είναι σημαντικό να σημειωθεί ότι επίδοση του ALVR εξαρτάται από τις δυνατότητες του υπολογιστή καθώς και από την ποιότητα του Wi-Fi δικτύου.

[3] https://github.com/alvr-org/ALVR

3.4 CloudXR



Εικόνα 10: Αρχιτεκτονική CloudXR

(Πηγή: https://www.nvidia.com/en-us/design-visualization/solutions/cloud-xr/)

Το CloudXR[4] είναι μια τεχνολογία η οποία δίνει την δυνατότητα μετάδοσης υψηλής πιστότητας περιεχομένου εικονικής πραγματικότητας καθώς και επαυξημένης πραγματικότητας από οποιαδήποτε OpenVR XR εφαρμογή σε έναν remote server-cloud.

Το CloudXR αξιοποιεί ισχυρούς cloud servers που χρησιμοποιούν κάρτες γραφικών υψηλής επίδοσης (NVDIA RTX) για να χειριστούν την διαδικασία του rendering, δηλαδή οι "βαριές" υπολογιστικές διεργασίες του rendering εκτελούνται στο cloud και όχι στην συσκευή του χρήστη. Μόλις το περιεχόμενο αποδοθεί στο cloud, το CloudXR συμπιέζει και κωδικοποιεί τα οπτικά και τα δεδομένα θέσης σε ένα stream που μπορεί να μεταδοθεί αποτελεσματικά μέσω δικτύων. Αυτό το stream γίνεται optimized για να μειώσει το latency και να διατηρεί υψηλή ποιότητα εικόνας. Στην συνέχεια οι χρήστες έχουν πρόσβαση στην VR ή AR εμπειρία χρησιμοποιώντας client συσκευές, όπως VR headset, smartphone ή tablet. Αυτές οι συσκευές λαμβάνουν το stream συμπιεσμένων δεδομένων από το cloud και το αποκωδικοποιούν για να ανασυνθέσουν τις οπτικές και πληροφορίες θέσης. Η συσκευή του client εμφανίζει το ανακατασκευασμένο περιεχόμενο VR ή AR στον χρήστη. Ανάλογα με τον τύπο της συσκευής, αυτό μπορεί να γίνει μέσω της οθόνης του VR headset ή της οθόνης της συσκευής. Οι χρήστες μπορούν να αλληλεπιδράσουν με το περιεχόμενο χρησιμοποιώντας συσκευές εισόδου, όπως χειριστήρια ή οθόνες αφής. Το CloudXR επιτρέπει σε πολλούς χρήστες να έχουν πρόσβαση στην ίδια VR ή AR εμπειρία ταυτόχρονα. Δεδομένου ότι η απόδοση πραγματοποιείται στο cloud, χρήστες με συσκευές διαφορετικών δυνατοτήτων μπορούν να έχουν πρόσβαση σε εμπειρίες υψηλής ποιότητας χωρίς την ανάγκη ισχυρού τοπικού hardware.

[4] https://www.nvidia.com/en-us/design-visualization/solutions/cloud-xr/

3.5 FusedVR

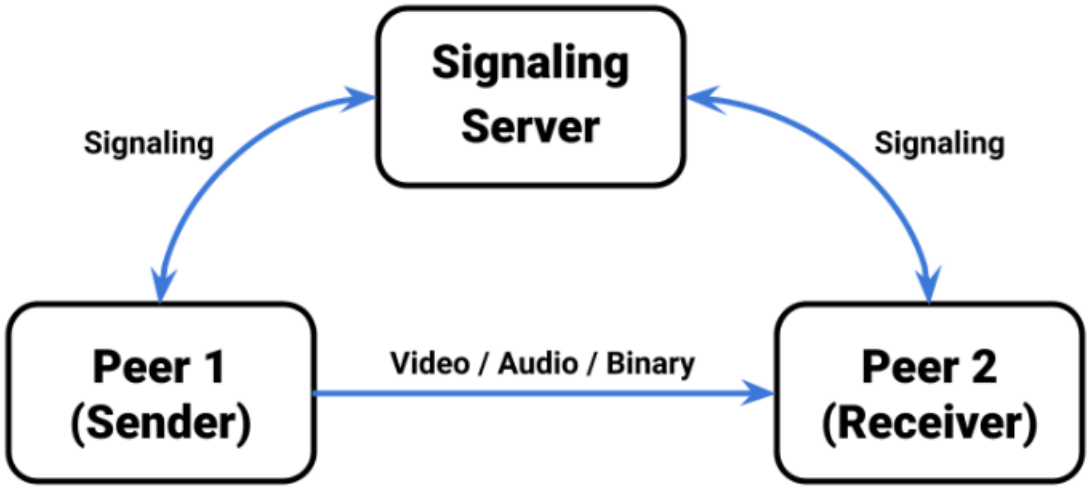
3.5.1 Render Streaming

Το Render Streaming[5] είναι ένα πακέτο της Unity το οποίο έχει χτιστεί πάνω στο WebRTC. Χρησιμοποιείται για να δημιουργεί γρήγορες streaming λύσεις peer to peer. Πιο συγκεκριμένα μπορεί να μεταδώσει ένα βίντεο το οποίο γίνεται rendered στην Unity σε ένα client μέσω του διαδικτύου. Επίσης μπορεί να κάνει stream ήχους που παράγονται στην Unity σε πολλαπλούς clients. Τέλος υποστηρίζει απομακρυσμένο χειρισμό δηλαδή μπορεί να στείλει μηνύματα Input (mouse,keyboard,touch,gamepad) στην Unity μέσω του client.

3.5.2 Δομή του Render Streaming

Το σύστημα Render Streaming αποτελείται από 3 μέρη:

- Signaling Server
- Peer 1 (Αποστολέας)
- Peer 2 (Δέκτης)



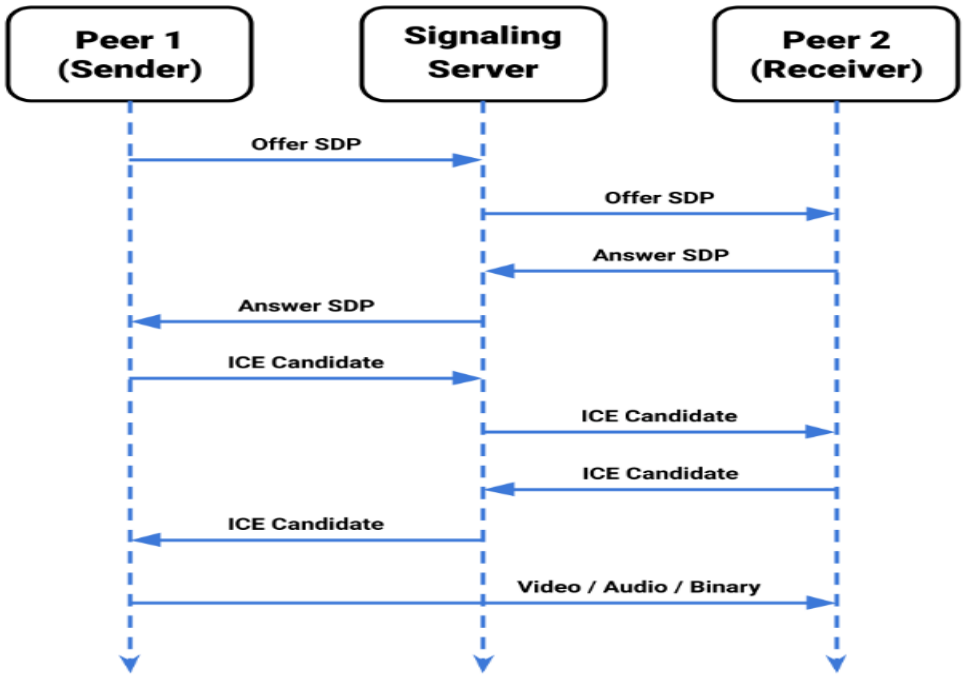
Εικόνα 11: Σύστημα Render Streaming

(Πηγή: https://docs.unity3d.com/Packages/com.unity.renderstreaming@3.1/manual/overview.html)

Στο Render Streaming δημιουργείται ένα δίκτυο P2P μεταξύ δύο peers και αυτό το δίκτυο στέλνει βίντεο/ήχο/δυαδικά δεδομένα. Ο Web server επιτρέπει την επικοινωνία μεταξύ δύο peers. Αυτή η επικοινωνία ονομάζεται signaling.

[5] https://docs.unity3d.com/Packages/com.unity.renderstreaming@3.1/manual/index.html

3.5.3 Η διεργασία Signaling



Εικόνα 12: Διεργασία Signaling

(Πηγή: https://docs.unity3d.com/Packages/com.unity.renderstreaming@3.1/manual/overview.html)

1 Το Peer 1 στέλνει Offer SDP στον Signaling server.

2 Το Peer 2 ελέγχει τον Signaling server για μη επεξεργασμένα Offer SDP και λαμβάνει ό,τι βρεθεί.

3 Το Peer 2 στέλνει το Answer SDP στο Signaling Server.

4 Το Peer 1 ελέγχει τον Signaling server για μη επεξεργασμένα Answer SDP και λαμβάνει όποια βρεθούν.

5 Το Peer 1 στέλνει το ICE Candidate στον Signaling server.

6 Το Peer 2 ελέγχει τον Signaling server για μη επεξεργασμένα ICE Candidate και λαμβάνει ό,τι βρεθεί.

7 Το Peer 2 στέλνει το ICE Candidate στον Signaling server.

8 Το Peer 1 ελέγχει τον Signaling server για μη επεξεργασμένο ICE Candidate και λαμβάνει ό,τι βρεθεί.

9 Η ίδρυση της σύνδεσης μεταξύ peers έχει ολοκληρωθεί και ο Peer 1 μπορεί να στείλει streams στο Peer 2.

3.5.4 Λειτουργία FusedVR

Το FusedVR[6] είναι ένα site το οποίο περιέχει το VR Render Streaming SDK (Software Development Kit). Αυτό το SDK έχει δημιουργηθεί από το συνδυασμό δύο εργαλείων:

1 Render Streaming

2 WebXR

Το FusedVR χρησιμοποιεί το Render Streaming για να μεταδώσει τα δεδομένα από δύο κάμερες της Unity σε ένα Web browser. Στην συνέχεια δημιουργείτε η στερεοσκοπική προβολή με τις δύο εικόνες, μία για το κάθε μάτι επίσης στέλνονται τα δεδομένα θέσης και στροφής του HMD καθώς και τα δεδομένα από τα χειριστήρια στην Unity. Η Unity χρησιμοποιεί αυτά τα δεδομένα για να ενημερώσει την σκηνή και στη συνέχεια στέλνει τα video streams στο browser τα οποία μπορούμε να δούμε μέσα στο VR. Το WebXR χρησιμοποιείται ώστε τα video stream να φθάσουν στο Web και να τα εμφάνιση μέσα στο VR χρησιμοποιώντας το WebXR API. Δηλαδή από μια σελίδα WebXR στο browser ενός HMD μπορούμε να δούμε αυτά τα video streams μέσα στο VR.

[6] https://fusedvr.com/

3.6 Εργαλεία συμπίεσης.

3.6.1 FFmpeg

Το FFmpeg[7] είναι ένα framework πολυμέσων το οποίο χρησιμοποιείται για επεξεργασία ήχου και βίντεο. Δίνει την δυνατότητα συμπιέσεις ενός video μειώνοντας τον αριθμό των bit rate ή χρησιμοποιώντας αλγόριθμους που ονομάζονται codec. Τα περισσότερα codec που χρησιμοποιεί έχουν απώλειες.

3.6.2 Draco

Το Draco[8] είναι ένα εργαλείο για συμπίεση και αποσυμπιέση τρισδιάστατων γεωμετρικών meshes και point clouds. Χρησιμοποιείται για την συμπίεση τρισδιάστατων μοντέλων με μεγάλο αριθμό κορυφών και υψηλά επίπεδα λεπτομέρειας τα οποία μπορεί να είναι δύσκολο να αποθηκευτούν και να μεταδοθούν χρησιμοποιώντας συνηθισμένες τεχνικές συμπίεσης.

[7] https://ffmpeg.org/
[8] https://google.github.io/draco/

Κεφάλαιο 4: Πείραμα

Στο πλαίσιο της παρούσας διπλωματικής εργασίας πραγματοποιήθηκε πειραματισμός και υλοποίηση μιας εφαρμογής remote rendering VR για φορητές συσκευές με σκοπό την διαχείριση υδατικών πόρων.

4.1 Εργαλεία Πειράματος

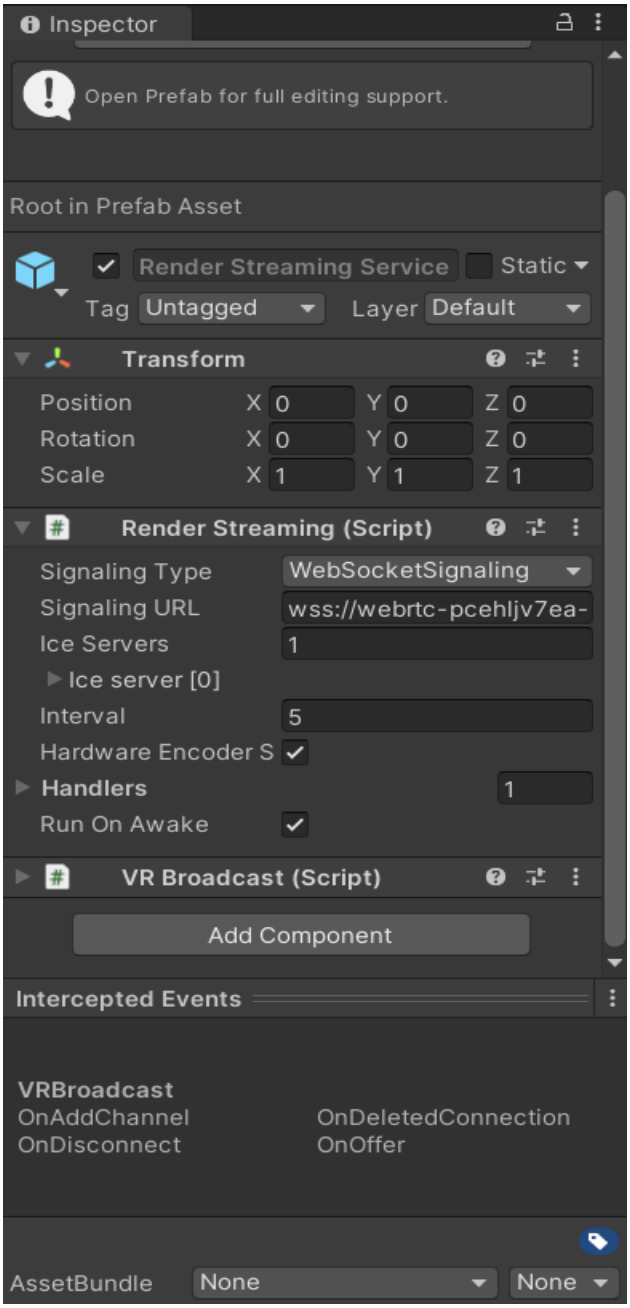
Για την εφαρμογή χρησιμοποιήθηκε η game engine Unity καθώς και το εργαλείο FusedVR. Η Unity περιέχει game objects (π.χ. κάμερα, 3D μοντέλα, αντικείμενα φωτισμού) τα οποία τοποθετούνται στην τρισδιάστατη σκήνη της. Ειδικότερα, τα game objects περιέχουν components (π.χ.scripts, transform) τα οποία δίνουν ιδιότητες σε αυτά. Όλα τα αντικείμενα περιέχουν το component transform (θέση, περιστροφή, κλίμακα). Το πακέτο FusedVR περιέχει δύο prefab (game objects που έχουν αποθηκευτεί με συγκεκριμένα components και μπορούν να επαναχρησιμοποιηθούν), το Render Streaming Service και το WebXR Streamer.

4.2 Λειτουργικότητα του Render Streaming Service

Στην εικόνα 13 διακρίνονται τα components του Render Streaming Service. Το Render Streaming Service περιέχει επιπλέον δύο scripts το RenderStreaming και το VRBroadcast.

RenderStreaming: Αυτό το script διαχειρίζεται το WebRTC video streaming στην Unity. Περιλαμβάνει λειτουργικότητα για τη διαμόρφωση Ice servers, για την δημιουργία και την διαχείριση ενός signaling server και την εγγραφή σε handlers για διάφορα events στην διαδικασία του WebRTC signaling. Επίσης δίνει την δυνατότητα για hardware encoding αν ο rendering server χρησιμοποιεί κάρτα γραφικών της Nvidia.

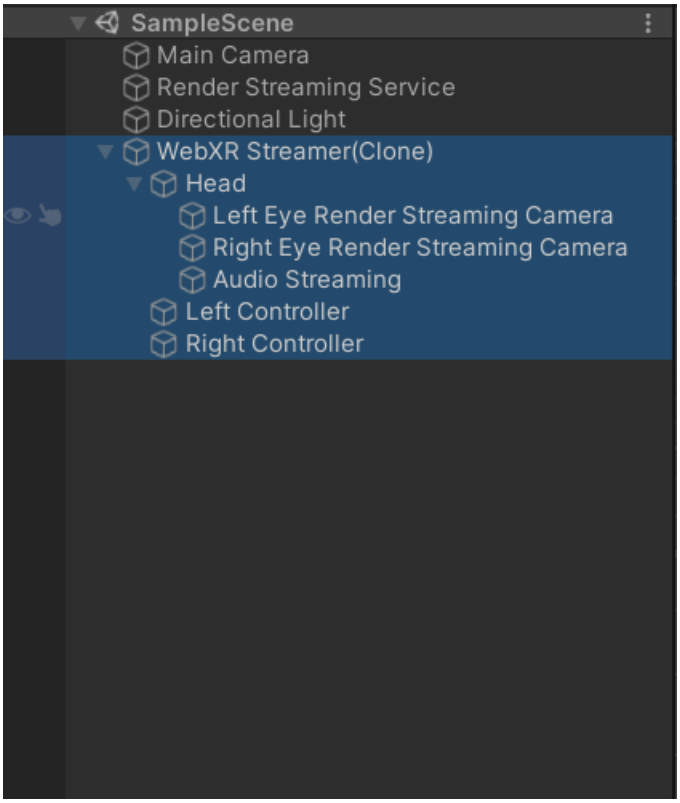
VRBroadcast: Το VRBroadcast script είναι υπεύθυνο για να ανταποκρίνεται στις προσφορές από τους client.



Εικόνα 13: Render Streaming Service

4.3 Λειτουργικότητα WebXR Streamer

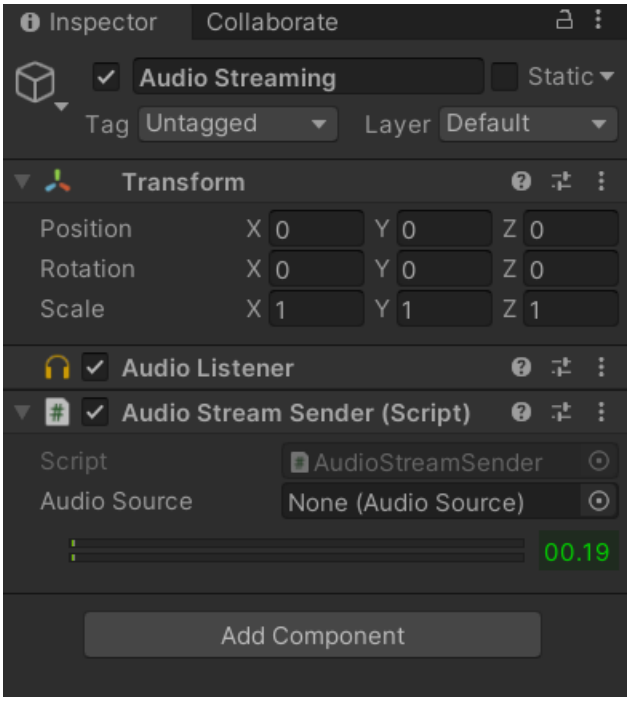
Το WebXR Streamer game object είναι parent object για τρία game objects (εικόνα 14). Πιο συγκεκριμένα συμπεριλαμβάνει ένα game object Head και δύο για τα controller (αριστερό και δεξί).



Εικόνα 14: WebXR Streamer

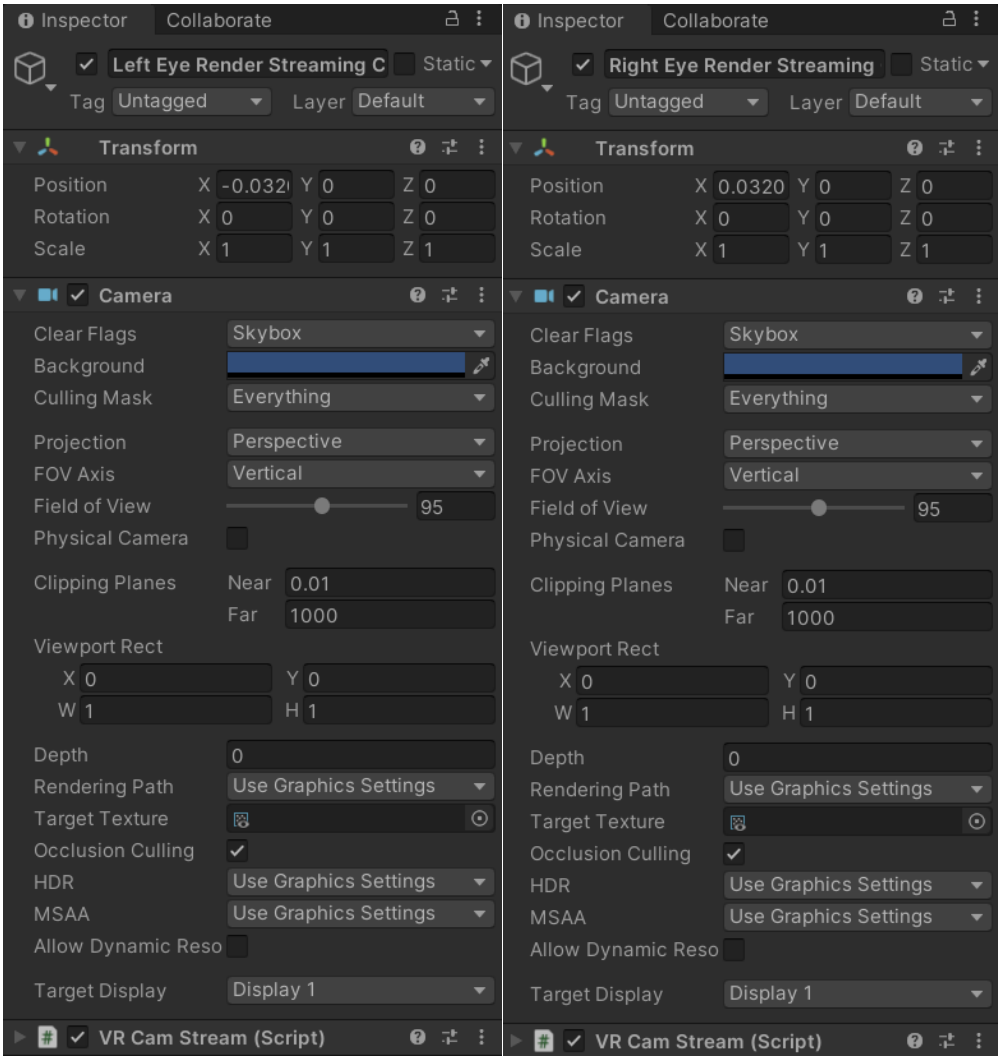
Το Head είναι parent object τριών game object, περιέχει δύο κάμερες μία για κάθε μάτι και ένα Audio Streaming.

Στην εικόνα 15 φαίνονται τα components του Audio Streaming. Περιέχει ένα Audio Listener και ένα Audio Stream Sender script. Το Audio Listener δίνει την δυνατότητα στον παίκτη να ακούσει ήχους από διάφορα Audio Source που βρίσκονται στον κόσμο του παιχνιδιού ενώ το Audio Stream Sender στέλνει τους ήχους που λαμβάνει στον client.



Εικόνα 15: Audio Streaming

Έπειτα, στις εικόνες 16 και 17 φαίνονται τα components των δύο καμερών που είναι υπεύθυνες για την στερεοσκοπική προβολή. Η κάμερα είναι το game object που καταγράφει τον κόσμο του παιχνιδιού και τον εμφανίζει στον παίχτη. Το component Camera μας δίνει την δυνατότητα να επεξεργαστούμε τις ιδιότητες της κάμερας. Το VR Cam Stream script καθορίζει διάφορες παραμέτρους των καμερών όπως την μέγιστη τιμή του frame rate, το anti aliasing, το bit rate καθώς και το ID στο οποίο θα συνδεθεί. Επίσης το script αυτό είναι υπεύθυνο για να δημιουργήσει VR stream από τις κάμερες. Το interpupillary distance είναι 64mm (IPD).



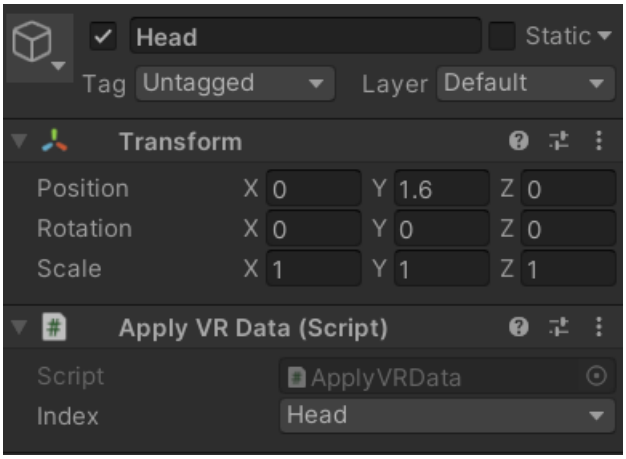
Εικόνα 16: Left Eye Render Streaming Camera-Εικόνα 17 Right Eye Render Streaming Camera

```
protected override MediaStreamTrack CreateTrack() {

    RenderTextureFormat format = WebRTC.GetSupportedRenderTextureFormat(SystemInfo.graphicsDeviceType);
    RenderTexture rt = new RenderTexture(streamingSize.x, streamingSize.y, depth, format) {
        antiAliasing = antiAliasing
    };
    rt.Create();

    // divide cameras into n sections over the canvas
    for ( int i = 0; i < cameras.Length; i++) {
        cameras[i].targetTexture = rt;
        cameras[i].rect = new Rect(new Vector2(i / cameras.Length, 0f), new Vector2(1 / cameras.Length, 1f));
    }

    return new VideoStreamTrack(rt);
}
```

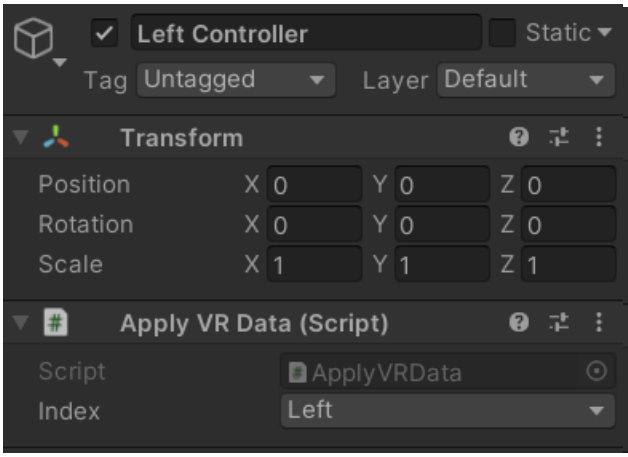
Εικόνα 18 Συνάρτηση CreateTrack

Στην εικόνα 18 φαίνεται ένα κομμάτι από τον κώδικα του script το οποίο πετυχαίνει το VR stream. Αρχικά αποκτάει ένα format για render texture σύμφωνα με το γραφικό API (OpenGL, Vulkan, DirectX) που χρησιμοποιεί η εφαρμογή. Στην συνέχεια δημιουργεί ένα render texture (τύπος texture που δημιουργείται και ενημερώνεται καθώς η Unity εκτελεί το πρόγραμμα) με βάση το format. Έπειτα το render texture αυτό γίνεται το target texture των καμερών το οποίο σημαίνει ότι αντί η κάμερα να κάνει render κατευθείαν στην οθόνη το κάνει σε ένα texture. Τέλος τοποθετεί τα αποτελέσματα των καμερών μέσα στο render texture το ένα δίπλα στο άλλο, αυτό δίνει την δίνει την δυνατότητα να συνδυαστούν τα αποτελέσματα των δύο καμερών σε ένα video stream.

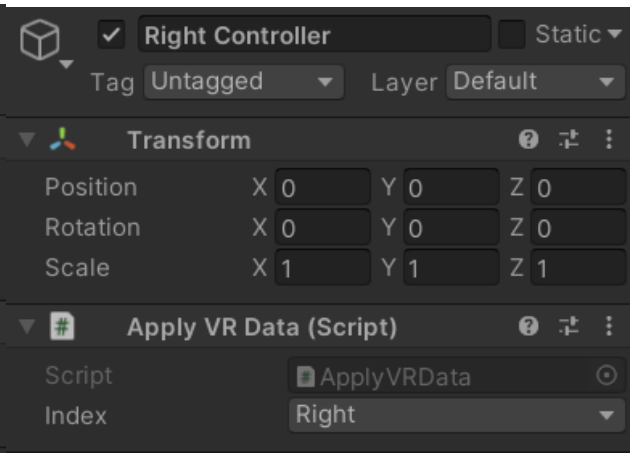
Τα game object Head, Left controller και Right Controller έχουν και τα τρία το Apply VR Data script στα components τους το οποίο εφαρμόζει τα δεδομένα θέσης και περιστροφής του VR από τον Client στον Server.



Εικόνα 19: Head

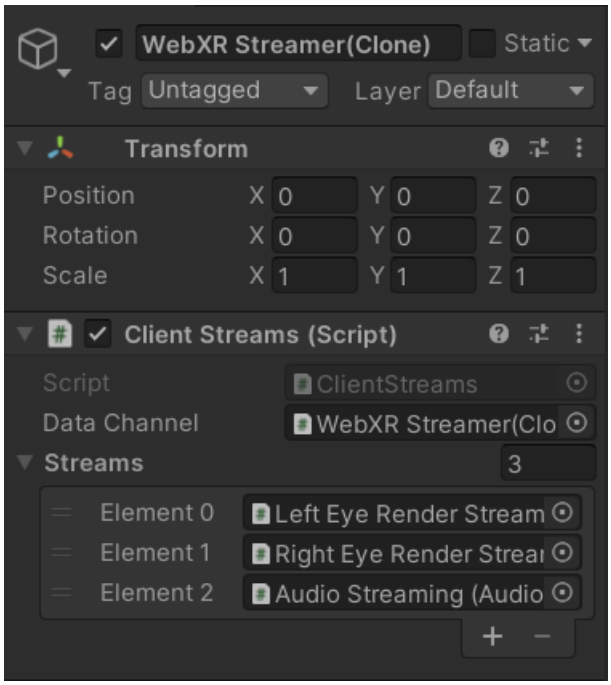


Εικόνα 20: Left Controller

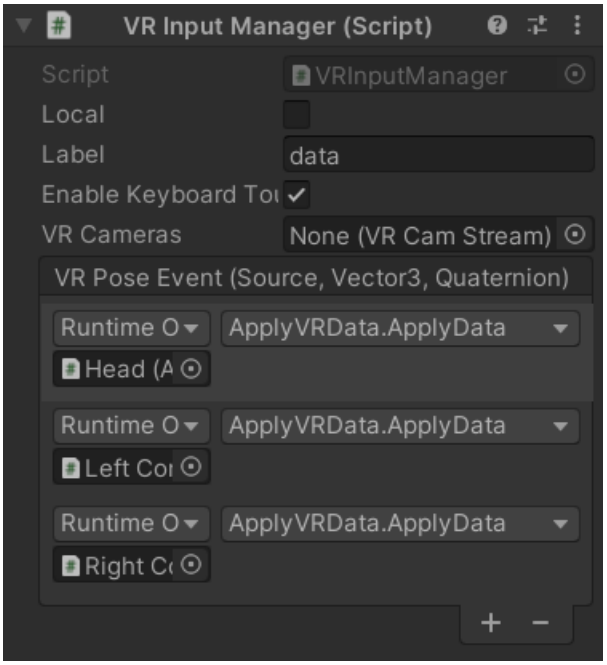


Εικόνα 21: Right Controller

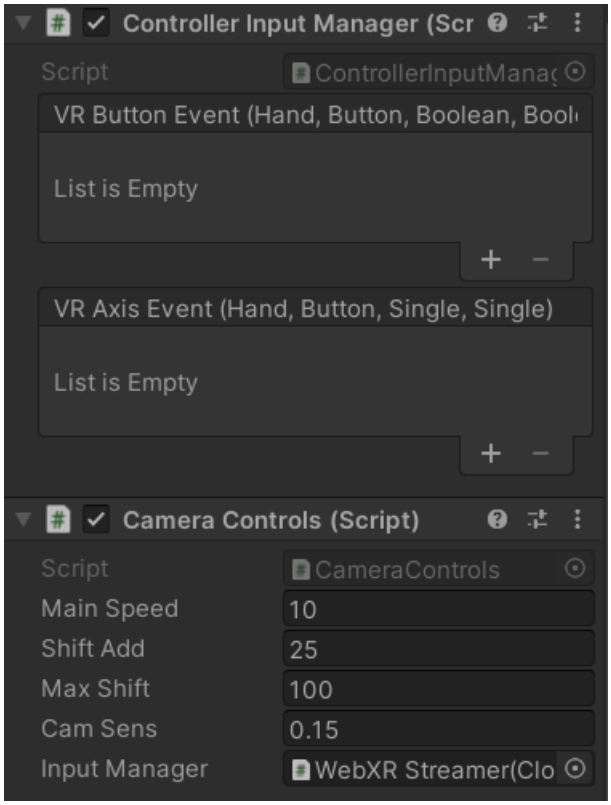
Στις παρακάτω εικόνες φαίνονται τα components του WebXR Streamer. Το WebXR Streamer περιέχει τέσσερα script. Το Client Streams, VR Input Manager, Controller Input Manager και Camera Controls.



Εικόνα 22: Client Streams script



Εικόνα 23: VR Input Manager script



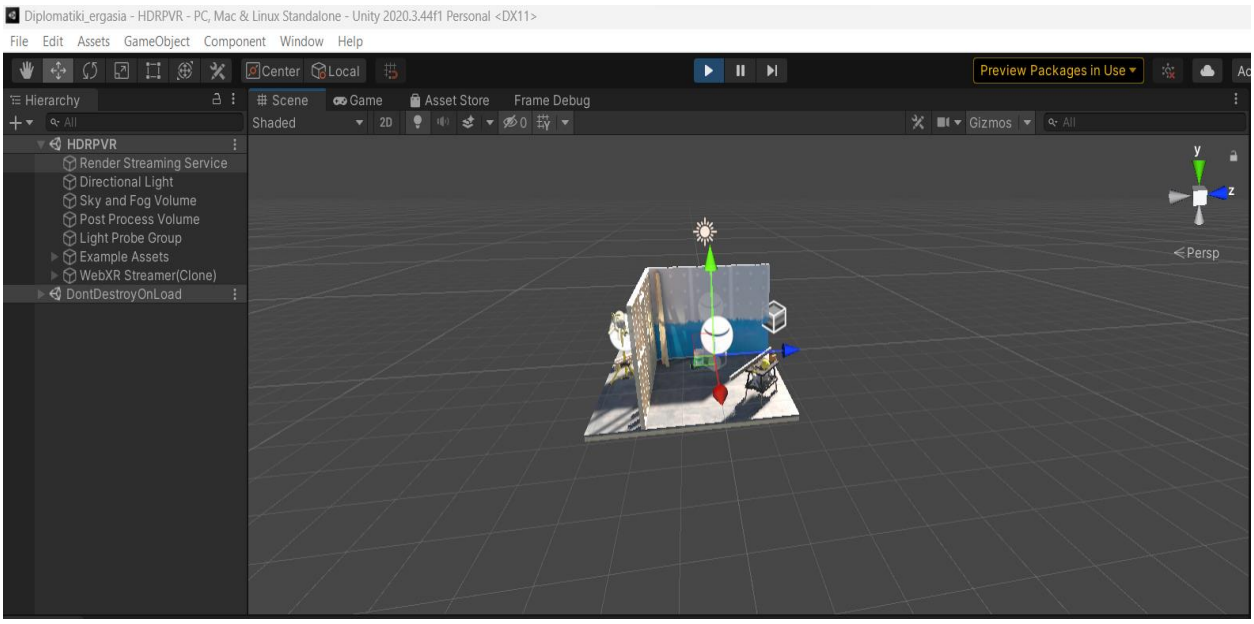
Εικόνα 24: Controller Input Manager και Camera Controls script

Το Client Streams είναι υπεύθυνο για την διαχείριση τον streams που θέλουμε να μεταδώσουμε στον Client. Παρατηρούμε στην εικόνα 22 ότι τα Stream που μεταδίδονται στον Client είναι οι κάμερες για το αριστερό και το δεξί μάτι καθώς και το Audio Streaming.

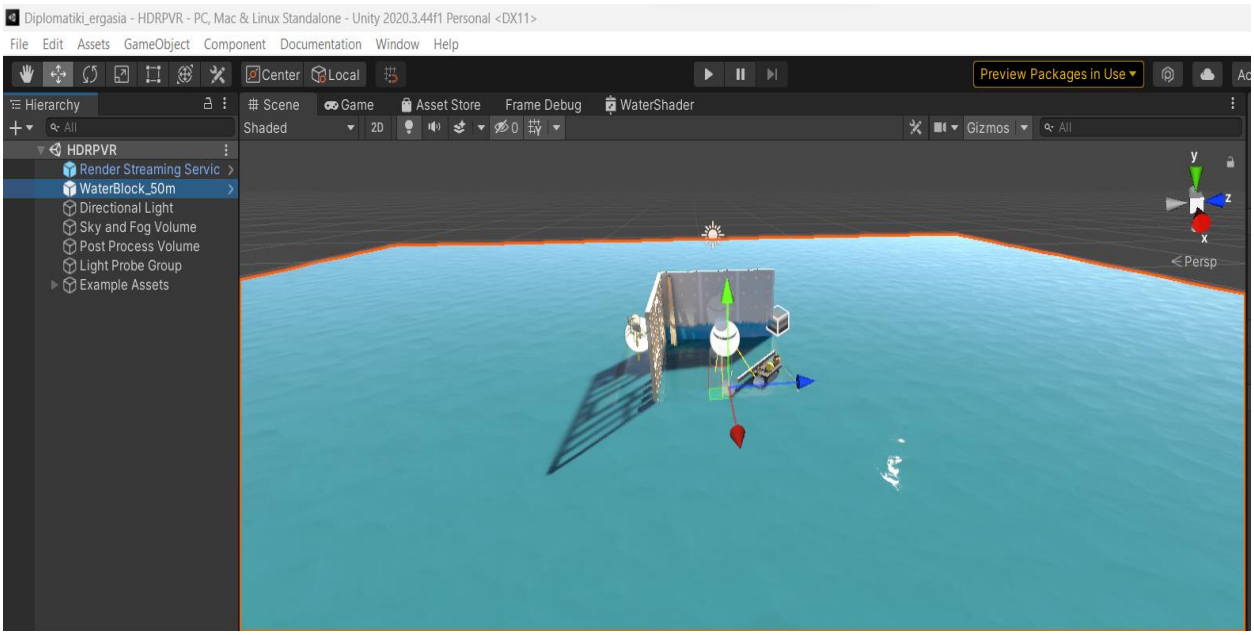
Το VRInputManager είναι υπεύθυνο για τον χειρισμό των δεδομένων από τον client. Πιο συγκεκριμένα δέχεται remote input από τον client και το ενσωματώνει στο Input system της unity. Επίσης αν χρησιμοποιείται VR headest δέχεται δεδομένα θέσης και περιστροφής του headset και των controllers. Στην περίπτωση που δεν υπάρχει VR headset συνδεδεμένο το script Camera Controls ενεργοποιείται και δίνει την δυνατότητα κίνησης της κάμερας στον client από το πληκτρολόγιο και το ποντίκι.

Τέλος το Controller Input Manager script διαχειρίζεται input data από τα controller όπως trigger, grips κτλ και τα μετατρέπει σε events.

4.4 Αποτελέσματα πειράματος

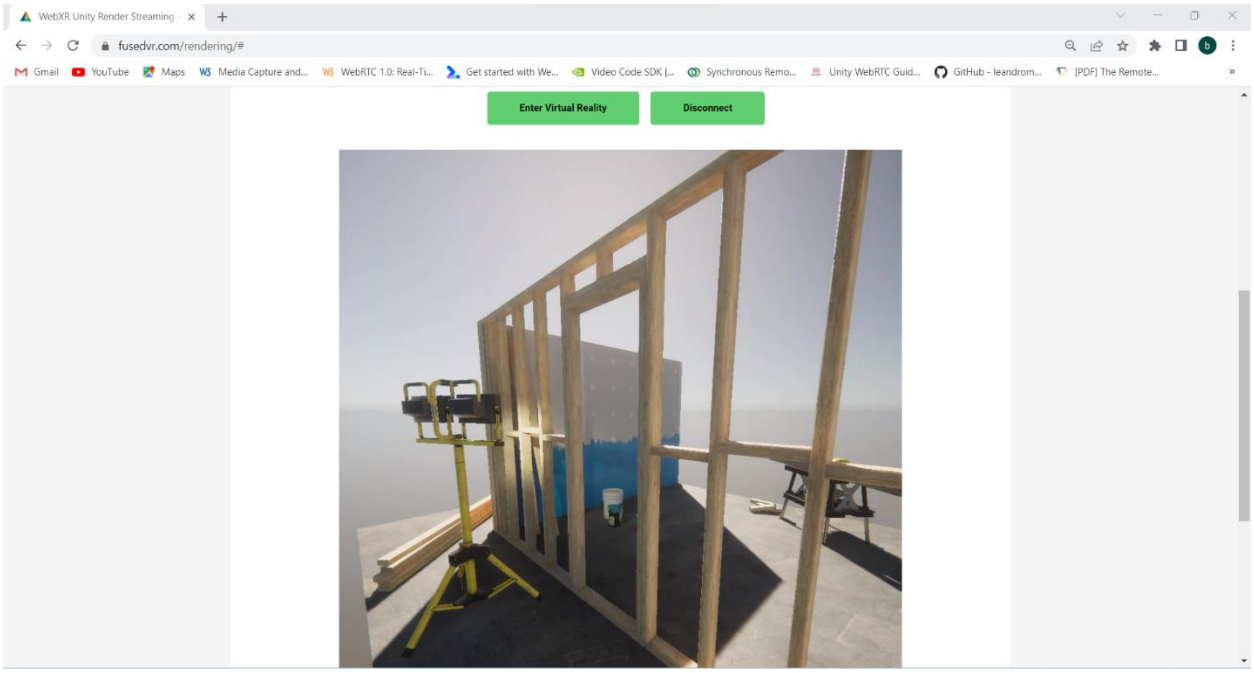


Εικόνα 25: Τρισδιάστατη σκήνη στην Unity

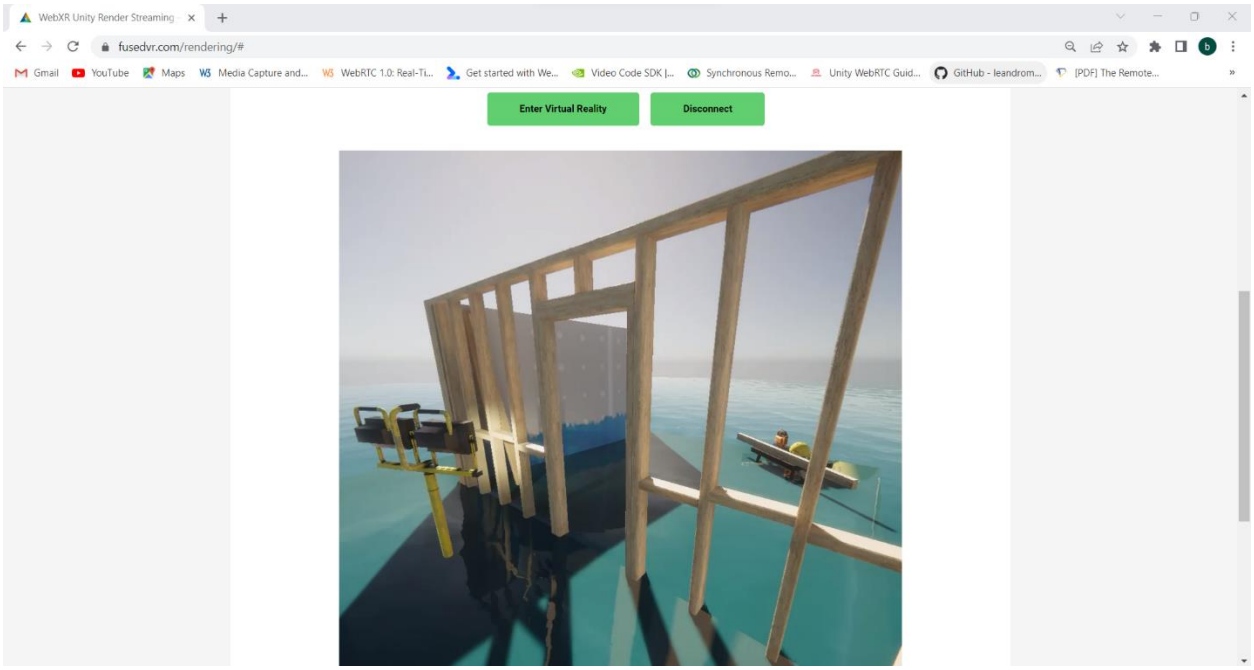


Εικόνα 26: Τρισδιάστατη πλημμυρισμένη σκήνη στην Unity

Στις εικόνες 25 και 26 παρουσιάζονται οι επιθυμητές σκηνές της Unity που μεταδίδονται.



Εικόνα 27: Μετάδοση εικόνας



Εικόνα 28 Μετάδοση πλημμυρισμένης εικόνας

Στην εικόνα 27 φαίνεται η σκήνη της Unity αφού μεταδοθεί από την Unity στον client από μια συγκεκριμένη οπτική γωνία. Αν ο χρήστης δεν χρησιμοποιεί κάποιο HMD μπορεί να μετακίνηση την κάμερα με πλήκτρα από το πληκτρολόγιο διαφορετικά μπορεί να επιλέξει το κουμπί “Enter Virtual Reality” για να αρχίσει να βλέπει στερεοσκοπικά μέσα από το HMD και να χρησιμοποιεί τα controllers για να αλληλεπίδραση με τον εικονικό κόσμο. Τέλος, στην εικόνα 28 παρουσιάζεται η σκήνη της unity έχοντας ενσωματώσει δεδομένα νερού.

4.5 Αξιολόγηση Πειράματος

Για την αξιολόγηση του πειράματος χρησιμοποιήθηκαν δύο κάρτες γραφικών από διαφορετικές εταιρίες, της AMD και της Nvidia καθώς οι κάρτες γραφικών της Nvidia δίνουν την δυνατότητα στον χρήστη να χρησιμοποιήσει hardware encoding το οποίο σημαίνει ότι οι κάρτες έχουν ειδικό hardware για την κωδικοποίηση των frames και δεν χρησιμοποιούν τους πυρήνες τους με αποτέλεσμα να αυξηθούν τα frame per second της εφαρμογής. Επίσης αξίζει να αναφερθεί ότι χρησιμοποιήθηκε για το πείραμα το HDRP (High Definition Rendering Pipeline) για την δημιουργία γραφικών υψηλής πιστότητας. Στην περίπτωση της κάρτας γραφικών της AMD που δεν χρησιμοποιείται hardware encoding τα fps κυμαίνονται από 27 μέχρι 33 ενώ με την κάρτα γραφικών της NVDIA κυμαίνονται από 52 μέχρι 57. Παρατηρούμε ότι ενώ η εφαρμογή πετυχαίνει την απομακρυσμένη απόδοση σε VR δεν το πετυχαίνει στην ιδανική περίπτωση η οποία θα ήταν 90 fps ώστε να ταιριάζει με τον ρυθμό ανανέωσης των περισσοτέρων HMD. Τα αποτελέσματα του πειράματος επηρεάζονται αισθητά από τον τύπο και την ταχύτητα του δικτύου σύνδεσης. Στην περίπτωση του παρόντος πειράματος η διαδικτυακή σύνδεση παρουσίαζε προβλήματα τα οποία είχαν άμεσο αντίκτυπο στο τελικό αποτελέσματα.

Κεφάλαιο 5: Συμπεράσματα-Προτάσεις

5.1 Συμπεράσματα

Η απομακρυσμένη απόδοση για VR παρουσιάζει πολλά πλεονεκτήματα που έχουν σημαντικές επιπτώσεις για το μέλλον της εικονικής πραγματικότητας. Με τη μεταφόρτωση του υπολογιστικού φόρτου εργασίας σε ισχυρούς απομακρυσμένους διακομιστές, η απομακρυσμένη απόδοση βελτιώνει την επίδοση, επιτρέποντας οπτικά εντυπωσιακές και καθηλωτικές εμπειρίες VR ακόμη και σε συσκευές με περιορισμένη υπολογιστική ισχύ. Επιπλέον, η απομακρυσμένη απόδοση προωθεί τη συνεργασία σε πραγματικό χρόνο και τις εμπειρίες πολλών παικτών στο VR, επιτρέποντας στους χρήστες να αλληλεπιδρούν μεταξύ τους ανεξάρτητα από τη φυσική τους τοποθεσία. Αυτό έχει εφαρμογές στα παιχνίδια, την εκπαίδευση και τις βιομηχανίες που απαιτούν απομακρυσμένη ομαδική εργασία. Ωστόσο, πρέπει να αντιμετωπιστούν προκλήσεις όπως οι περιορισμοί του bandwidth και τα ζητήματα του latency για να παρέχεται μια απρόσκοπτη εμπειρία απομακρυσμένης απόδοσης. Επιπλέον, πρέπει να εφαρμόζονται μέτρα ασφάλειας δεδομένων για την προστασία των πληροφοριών του χρήστη κατά τη μετάδοση δεδομένων σε απομακρυσμένους διακομιστές. Συνολικά, η απομακρυσμένη απόδοση για VR ανοίγει νέες δυνατότητες και προόδους σε αυτόν τον τομέα. Έχει τη δυνατότητα να καταστήσει τις εμπειρίες VR υψηλής ποιότητας προσβάσιμες σε ένα ευρύτερο κοινό, να ενθαρρύνει τη συνεργασία και τις αλληλεπιδράσεις πολλών παικτών. Αντιμετωπίζοντας τις προκλήσεις και διασφαλίζοντας την ασφάλεια των δεδομένων, η απομακρυσμένη απόδοση μπορεί να φέρει επανάσταση στο τοπίο της εικονικής πραγματικότητας και να ανοίξει το δρόμο για ένα πιο καθηλωτικό και διασυνδεδεμένο μέλλον εικονικής πραγματικότητας.

5.2 Προτάσεις

Για την βελτίωση του εργαλείου που υλοποιήθηκε στην διπλωματική εργασία προτείνονται οι παρακάτω ενέργειες:

1) Η επίδοση της εφαρμογής εξαρτάται σε μεγάλο βαθμό από την κάρτα γραφικών που χρησιμοποιεί ο rendering server. Αν ο rendering server χρησιμοποιεί κάρτα γραφικών της AMD, αυτό έχει ως αποτέλεσμα να τρέχει περίπου σε 30 fps τα οποία είναι χαμηλά ενώ στην περίπτωση που χρησιμοποιεί κάρτα της NVDIA τρέχει περίπου σε 60. Τα 60 fps θεωρούνται αποδεκτά για εφαρμογές που δεν χρησιμοποιούν VR και στην περίπτωση της εικονικής πραγματικότητας θεωρείται ο ελάχιστος αριθμός. Η ιδανική περίπτωση για εφαρμογές εικονικής πραγματικότητας είναι πάνω από 90 fps για την πιο ομαλή εμπειρία του χρήστη και την αποφυγή του motion sickness. Οπότε η εφαρμογή θα πρέπει να γίνει optimized ώστε ο χρήστης να μπορεί να έχει την πιο ομαλή εμπειρία εικονικής πραγματικότητας ανεξάρτητα από την κάρτα γραφικών που χρησιμοποιεί ο rendering server.
2) Η εφαρμογή αυτή δημιουργήθηκε ώστε να μπορούμε να χρησιμοποιήσουμε ασύρματο HMD καθώς και να μπορούμε να ελέγξουμε ένα υδραυλικό έργο ή ένα κτήριο το οποίο έχει υποστεί ζημία από πλημμύρα χωρίς να βρισκόμαστε

στην τοποθεσία αυτή. Για τον λόγο αυτό θα πρέπει να χρησιμοποιηθούν δεδομένα σε πραγματικό χρόνο, οτιδήποτε άλλο θα θεωρηθεί μια προσομοίωση της κατάστασης η οποία μπορεί να μην είναι αντιπροσωπευτική. Καθοριστικό ρόλο για την βελτίωση του συγκεκριμένου ζητήματος διαδραματίζει ο τομέας της φωτογραμμετρίας. Ο συγκεκριμένος κλάδος εξελίσσεται ραγδαία και διεισδύει σε ποικίλους τομείς όπως η γεωργία, δασοκομία, κτηνοτροφία, χωροταξία ακόμα και στη διαχείριση φυσικών φαινομένων μέσω εναέριας επισκόπησης με χρήση συστημάτων μη επανδρωμένων αεροσκαφών (ΣμηΕΑ). Τα ΣμηΕΑ συμβάλουν στην χαρτογράφηση γεωγραφικών δεδομένων με γρήγορο και οικονομικό τρόπο ακόμα και σε περιοχές που δεν ευνοείται η προσβασιμότητα.

Κεφάλαιο 6: Βιβλιογραφικές αναφορές

ALREGIB, G. AND ALTUNBASAK, Y. 2005. 3tp: An application-layer protocol for streaming 3-d models. Multimedia, IEEE Transactions on 7, 6, 1149–1156.

BEERMANN, D. AND HUMPHREYS, G. 2003. Visual computing in the future: Computer graphics as a remote service. University of Virginia, Computer Science Department, University of Virginia Technical Report CS-2003-16 25.

BARATTO, R. A., KIM, L. N., AND NIEH, J. 2005. Thinc: a virtual display architecture for thin-client computing. In Proceedings of the twentieth ACM symposium on Operating systems principles. SOSP '05. ACM, New York, NY, USA, 277–290.

BAO, P. AND GOURLAY, D. 2004. Remote walkthrough over mobile networks using 3-d image warping and streaming. Vision, Image and Signal Processing, IEE Proceedings - 151, 4, 329 – 336.

BEIGBEDER, T., COUGHLAN, R., LUSHER, C., PLUNKETT, J., AGU, E., AND CLAYPOOL, M. 2004. The effects of loss and latency on user performance in unreal tournament 2003. In Proc. of NetGames'04. Portland, OR, 144–151.

BERLIOS. 2008. Freenx - nx components. http://openfacts2.berlios.de/wikien/index.php/Berlios Project:FreeNX - NX Components

BOUKERCHE, A. AND PAZZI, R. W. N. 2006. Remote rendering and streaming of progressive panoramas for mobile devices. In Proceedings of the 14th annual ACM international conference on Multimedia. MULTIMEDIA '06. ACM, New York, NY, USA, 691–694.

CUMBERLAND, B. C., CARIUS, G., AND MUIR, A. 1999. Microsoft windows nt server 4.0 terminal server edition technical reference. Microsoft Press.

COMMANDER, D. R. 2007. Virtualgl: 3d without boundaries the virtualgl project. http://www.virtualgl.org/

CHANG, C.-F. AND GER, S.-H. 2002. Enhancing 3d graphics on mobile devices by image-based rendering. In Proceedings of the Third IEEE Pacific Rim Conference on Multimedia: Advances in Multimedia Information Processing. PCM '02. Springer-Verlag, London, UK, 1105–1111.

CAMPBELL, G., DEFANTI, T. A., FREDERIKSEN, J., JOYCE, S. A., AND LESKE, L. A. 1986. Two bit/pixel full color encoding. SIGGRAPH Comput. Graph. 20, 215–223.

COHEN-OR, D., CHRYSANTHOU, Y. L., SILVA, C. T., AND DURAND, F. 2003. A survey of visibility for walkthrough applications. Visualization and Computer Graphics, IEEE Transactions on 9, 3, 412–431.

CHEN, S. E. 1995. Quicktime vr: an image-based approach to virtual environment navigation. In Proceedings of the 22nd annual conference on Computer graphics and interactive techniques. SIGGRAPH '95. ACM, New York, NY, USA, 29–38.

CHEN, J., YOON, I., AND BETHEL, W. 2008. Interactive, internet delivery of visualization via structured prerendered multiresolution imagery. IEEE Trans. Vis. Comput. Graph. 14, 2, 302–312.

DIEPSTRATEN, J., GORKE, M., AND ERTL, T. 2004. Remote line rendering for mobile devices. In Computer Graphics International, 2004. Proceedings. IEEE, 454–461.

DUGUET, F. AND DRETTAKIS, G. 2004. Flexible point-based rendering on mobile devices. IEEE Comput. Graph. Appl. 24, 57–63.

D'AMORA, B. AND BERNARDINI, F. 2003. Pervasive 3d viewing for product data management. IEEE Comput. Graph. Appl. 23, 14–19.

DHARMAPURIKAR, M. 2013b. Method and mechanism for performing both server-side and client-side rendering of visual data. US Patent App. 13/349,422.

DU, Q. AND FOWLER, J. 2007. Hyperspectral image compression using JPEG2000 and principal component analysis. IEEE Geoscience and Remote Sensing Letters 4, 2, 201–205.

DE WINTER, D., SIMOENS, P., DEBOOSERE, L., DE TURCK, F., MOREAU, J., DHOEDT, B., AND DEMEESTER, P. 2006. A hybrid thin-client protocol for multimedia streaming and interactive gaming applications. In Proceedings of the 2006 international workshop on Network and operating systems support for digital audio and video. NOSSDAV '06. ACM, New York, NY, USA, 15:1–15:6.

DEERING, M. 1995. Geometry compression. In Proc. of the 22nd Annual Conference on Computer Graphics and Interactive Techniques (SIGGRATH'95). Los Angeles, CA, 13–20.

EGGER, O., LI, W., AND KUNT, M. 1995. High compression image coding using an adaptive morphological subband decomposition. IEEE 83, 2, 272–287.

EISERT, P. AND FECHTELER, P. 2008. Low delay streaming of computer graphics. In Image Processing, 2008. ICIP 2008. 15th IEEE International Conference on. 2704 – 2707.

FUNKHOUSER, T. A. 1995. Ring: a client-server system for multi-user virtual environments. In Proceedings of the 1995 symposium on Interactive 3D graphics. I3D '95. ACM, New York, NY, USA, 85–ff.

GAILLY, J. 1992. The gzip home page. http://www.gzip.org

GUMHOLD, S. AND STRABER, W. 1998. Real time compression of triangle mesh connectivity. In Proc. of the 25th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRATH'98). Orlando, FL, 133–140.

HUMPHREYS, G., HOUSTON, M., NG, R., FRANK, R., AHERN, S., KIRCHNER, P. D., AND KLOSOWSKI, J. T. 2002. Chromium: A stream-processing framework for interactive rendering on clusters. ACM Trans. Graph. 21, 3, 693–702.

HERZOG, R., KINUWAKI, S., MYSZKOWSKI, K., AND SEIDEL, H. 2008. Render2mpeg: A perception-based framework towards integrating rendering and video compression. In Computer Graphics Forum. Vol. 27. Wiley Online Library, 183–192.

HUDSON, T. C. AND MARK, W. R. 1999. Multiple image warping for remote display of rendered images. Tech. rep., Chapel Hill, NC, USA.

HUANG, C., HSU, C., CHANG, Y., AND CHEN, K. 2013. GamingAnywhere: An open cloud gaming system. In Proc. of the ACM Multimedia Systems Conference (MMSys'13). Oslo, Norway.

HESINA, G. AND SCHMALSTIEG, D. 1998. A network architecture for remote rendering. In DIS-RT. IEEE Computer Society, 88–91.

HEGE, H.-C., MERZKY, A., AND ZACHOW, S. 2000. Distributed visualization with opengl vizserver: Practical experiences. Tech. Rep. 00-31, ZIB, Takustr.7, 14195 Berlin.

IKONOMOPOULOS, A. AND KUNT, M. 1985. High compression image coding via directional filtering. Elsevier Signal Processing 8, 2, 179–203.

JURGELIONIS, A., FECHTELER, P., EISERT, P., BELLOTTI, F., DAVID, H., LAULAJAINEN, J. P., CARMICHAEL, R., POULOPOULOS, V., LAIKARI, A., PERAL¨ A¨ , P., DE GLORIA, A., AND BOURAS, C. 2009. Platform for distributed 3d gaming. Int. J. Comput. Games Technol. 2009, 1:1–1:15.

KOLLER, D., TURITZIN, M., LEVOY, M., TARINI, M., CROCCIA, G., CIGNONI, P., AND SCOPIGNO, R. 2004. Protected interactive 3d graphics via remote rendering. ACM Trans. Graph. 23, 695–703.

KUNT, M., BENARD, M., AND LEONARDI, R. 1987. Recent results in high-compression image coding. IEEE Transactions on Circuits and Systems 34, 11, 1306–1336.

LEVOY, M. 1995. Polygon-assisted jpeg and mpeg compression of synthetic images. In Proceedings of the 22nd annual conference on Computer graphics and interactive techniques. SIGGRAPH '95. ACM, New York, NY, USA, 21–28.

LAMBERTI, F., ZUNINO, C., SANNA, A., ANTONINO, F., AND MANIEZZO, M. 2003. An accelerated remote graphics architecture for pdas. In Proceedings of the eighth international conference on 3D Web technology. Web3D '03. ACM, New York, NY, USA, 55–ff.

LAMBERTI, F. AND SANNA, A. 2007. A streaming-based solution for remote visualization of 3D graphics on mobile devices. IEEE Trans. Vis. Comput. Graph. 13, 2, 247–260.

LLUCH, J., GAITAN´ , R., CAMAHORT, E., AND VIVO´ , R. 2005. Interactive three-dimensional rendering on mobile computer devices. In Proceedings of the 2005 ACM SIGCHI International Conference on Advances in computer entertainment technology. ACE '05. ACM, New York, NY, USA, 254–257.

LEE, Y. AND SONG, B. 2008. An intra-frame rate control algorithm for ultra low delay H.264/AVC coding. In Proc. of ICASSP'08. Las Vegas, NV, 1041–1044.

LIN, T. AND HAO, P. 2005. Compound image compression for real-time computer screen image transmission. IEEE Transactions on Image Processing 14, 8, 993–1005.

LI, J. AND KUO, C. 1998. A dual graph approach to 3D triangular mesh compression. In International Conference on Image Processing (ICIP'98). Chicago, Illinois, 891–894.

LI, B., WANG, Z., LIU, J., AND ZHU, W. 2013. Two decades of internet video streaming: A retrospective view. ACM Trans. Multimedia Comput. Commun. Appl. 9, 1s, 33:1–33:20.

LIU L., ZHONG R., ZHANG W., LIU Y., ZHANG J., ZHANG L., GRUTESER M. 2018. Cutting the cord.

MANN, Y. AND COHEN-OR, D. 1997. Selective pixel transmission for navigating in remote virtual environments. Comput. Graph. Forum 16, 3, 201–206.

MORVAN, Y., FARIN, D., AND WITH, P. 2007. Multiview depth-image compression using an extended h.264 encoder. In Springer Advances in Image and Video Technology. Delft, The Netherlands, 675–686.

MCMILLAN, L. 1997. An image-based approach to three dimensional computer graphics. Ph.D. thesis, University of North Carolina at Chapel Hill, Department of Computer Science.

MESSAOUDI F., KSENTINI A., SIMON G. 2015. Dissecting Games Engines: the Case of Unity3D

MENG J, PAUL S, HU C. 2020. Exploiting Frame Similarity to Enable High-Quality Multiplayer VR on Commodity Mobile Devices.

MARK, W. 1999. Post-rendering 3D image warping: Visibility, reconstruction, and performance for depthimage warping. Ph.D. thesis, University of North Carolina at Chapel Hill, Department of Computer Science.

MA, K.-L. AND CAMP, D. M. 2000. High performance visualization of time-varying volume data over a widearea network status. In Proceedings of the 2000 ACM/IEEE conference on Supercomputing (CDROM). Supercomputing '00. IEEE Computer Society, Washington, DC, USA.

MILANI, S. AND CALVAGNO, G. 2010. A cognitive approach for effective coding and transmission of 3d video. In Proceedings of the international conference on Multimedia. MM '10. ACM, New York, NY, USA, 581– 590.

MAHESWARI, D. AND RADHA, V. 2010. Enhanced layer based compound image compression. In Proc. of the Amrita ACM-W Celebration on Women in Computing in India (A2CWiC'10). Tamilnadu, India, 40:1– 40:8.

MARCELLIN, M. W., BILGIN, A., GORMISH, M. J., AND BOLIEK, M. P. 2000. An overview of jpeg-2000. In Proceedings of the Conference on Data Compression. DCC '00. IEEE Computer Society, Washington, DC, USA, 523–.

NVIDIA. 2009. Reality server. http://www.nvidia.com/object/realityserver.html.

NIEH, J., YANG, S. J., AND NOVIK, N. 2003. Measuring thin-client performance using slow-motion benchmarking. ACM Trans. Comput. Syst. 21, 87–115.

NOIMARK, Y. AND COHEN-OR, D. 2003. Streaming scenes to mpeg-4 video-enabled devices. IEEE Comput. Graph. Appl. 23, 58–64.

NAVE, I., DAVID, H., SHANI, A., TZRUYA, Y., LAIKARI, A., EISERT, P., AND FECHTELER, P. 2008. Games@large graphics streaming architecture. In Consumer Electronics, 2008. ISCE 2008. IEEE International Symposium on. 1 –4.

NADEEM, M., WONG, S., AND KUZMANOV, G. 2010. An efficient realization of forward integer transform in H.264/AVC intra-frame encoder. In Proc. of SAMOS'10. Samos, Greece, 71–78.

OHAZAMA, C. 1999. Opengl vizserver white paper. Silicon Graphics, Inc.

OH, H. AND HO, Y. 2006. H.264-based depth map sequence coding using motion information of corresponding texture video. In Springer Advances in Image and Video Technology. Hsinchu, Taiwan, 898–907.

OBERHUMER, M. 1996. Lzo – a real-time data compression library. http://www.oberhumer.com/opensource/lzo/

PANTEL, L. AND WOLF, L. C. 2002. On the suitability of dead reckoning schemes for games. In Proceedings of the 1st workshop on Network and system support for games. ACM, 79–84.

POPESCU, V., EYLES, J., LASTRA, A., STEINHURST, J., ENGLAND, N., AND NYLAND, L. 2000. The warpengine: An architecture for the post-polygonal age. In Proceedings of the 27th annual conference on Computer graphics and interactive techniques. ACM Press/Addison-Wesley Publishing Co., 433–442.

PERLMAN, S. G., LAAN, R. V. D., COTTER, T., FURMAN, S., MCCOOL, R., AND BUCKLEY, I. 2010. System and method for multi-stream video compression using multiple encoding formats. US Patent No. 2010/0166068A1.

PHIL KARLTON, PAULA WOMACK, J. L. 2005. Opengl graphics with the x window system (version 1.4). http://www.opengl.org/documentation/specs/.

PARAVATI, G., SANNA, A., LAMBERTI, F., AND CIMINIERA, L. 2011. An adaptive control system to deliver interactive virtual environment content to handheld devices. Mobile Networks and Applications 16, 3, 385–393

ROSS, P. 2009. Cloud computing's killer app: Gaming. IEEE Spectrum 46, 3, 14.

RICHARDSON, T., STAFFORD-FRASER, Q., WOOD, K. R., AND HOPPER, A. 1998. Virtual network computing. IEEE Internet Computing 2, 33–38.

REDERT, A., DE BEECK, M. O., FEHN, C., IJSSELSTEIJN, W., POLLEFEYS, M., GOOL, L. J. V., OFEK, E., SEXTON, I., AND SURMAN, P. 2002. Attest: Advanced three-dimensional television system technologies. In 3DPVT. IEEE Computer Society, 313–319.

SCHMIDT, B. K., LAM, M. S., AND NORTHCUTT, J. D. 1999. The interactive performance of slim: a stateless, thin-client architecture. In Proceedings of the seventeenth ACM symposium on Operating systems principles. SOSP '99. ACM, New York, NY, USA, 32–47.

SHADE, J., GORTLER, S., HE, L.-W., AND SZELISKI, R. 1998. Layered depth images. In Proceedings of the 25th annual conference on Computer graphics and interactive techniques. SIGGRAPH ’98. ACM, New York, NY, USA, 231–242.

SERRAL-GRACIA` , R., CERQUEIRA, E., CURADO, M., YANNUZZI, M., MONTEIRO, E., AND MASIP-BRUIN, X. 2010. An overview of quality of experience measurement challenges for video applications in ip networks. In Wired/Wireless Internet Communications. Springer, 252–263.

SHI, S., NAHRSTEDT, K., AND CAMPBELL, R. H. 2008. View-dependent real-time 3d video compression for mobile devices. In Proceeding of the 16th ACM international conference on Multimedia. MM ’08. ACM, New York, NY, USA, 781–784.

SHI, S., JEON, W. J., NAHRSTEDT, K., AND CAMPBELL, R. H. 2009. Real-time remote rendering of 3d video for mobile devices. In Proceedings of the 17th ACM international conference on Multimedia. MM ’09. ACM, New York, NY, USA, 391–400.

SHI, S., KAMALI, M., NAHRSTEDT, K., HART, J. C., AND CAMPBELL, R. H. 2010. A high-quality low-delay remote rendering system for 3d video. In Proceedings of the international conference on Multimedia. MM ’10. ACM, New York, NY, USA, 601–610.

SHI S. 2015. A Survey of Interactive Remote Rendering Systems

SHI, S., HSU, C.-H., NAHRSTEDT, K., HART, J. C., AND CAMPBELL, R. H. 2011a. Using graphics rendering contexts to enhance the real-time video coding for mobile cloud gaming. In Proceedings of the international conference on Multimedia. MM ’11.

SINGHAL, S. AND ZYDA, M. 1999. Networked virtual environments: design and implementation. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA.

SCHMALSTIEG, D. 1997. The remote rendering pipeline - managing geometry and bandwidth in distributed virtual environments. Ph.D. thesis, Institute of Computer Graphics and Algorithms, Vienna University of Technology, Favoritenstrasse 9-11/186, A-1040 Vienna, Austria.

SHI, S., NAHRSTEDT, K., AND CAMPBELL, R. 2012a. A real-time remote rendering system for interactive mobile graphics. ACM Transactions on Multimedia Computing, Communications, and Applications (TOMCCAP) 8, 3s, 46.

SEWARD, J. 1996. bzip2 and libbzip2, version 1.0.5: A program and library for data compression. http://www.bzip.org

SKODRAS, A., CHRISTOPOULOS, C., AND EBRAHIMI, T. 2001. The JPEG 2000 still image compression standard. IEEE Signal Processing Magazine 18, 5, 36–58.

SCHAFER, R. 1998. MPEG-4: a multimedia compression standard for interactive applications and services. Electronics and Communication Engineering Journal 10, 6, 253–262.v

TOUCH, J. 1995. Defining high-speed protocols: five challenges and an example that survives the challenges. Selected Areas in Communications, IEEE Journal on 13, 5, 828–835.

THOMAS, G., POINT, G., AND BOUATOUCH, K. 2005. A client-server approach to image-based rendering on mobile terminals. Tech. Rep. RR-5447, INRIA. January.

VETRO, A., WIEGAND, T., AND SULLIVAN, G. 2011. Overview of the stereo and multiview video coding extensions of the h.264/mpeg-4 avc standard. Proceedings of the IEEE 99, 4, 626 –642.

WANG, S. AND DEY, S. 2009. Modeling and characterizing user experience in a cloud server based mobile gaming approach. In Global Telecommunications Conference, 2009. GLOBECOM 2009. IEEE. IEEE, 1–7.

WANG, S. AND LIN, T. 2009. A unified LZ and hybrid coding for compound image partial-lossless compression. In Proc. of International Congress on Image and Signal Processing (CISP'09). 1–5.

WANG, S. AND LIN, T. 2012. United coding method for compound image compression. Multimedia Tools and Application.

WOODWARD, C., VALLI, S., HONKAMAA, P., AND HAKKARAINEN, M. 2002. Wireless 3d cad viewing on a pda device. In Proceedings of the 2nd Asian International Mobile Computing Conference (AMOC 2002). Vol. 14. 17.

WIEGAND, T., SULLIVAN, G., BJNTEGAARD, G., AND LUTHRA, A. 2003. Overview of the H.264/AVC video coding standard. IEEE Transactions on Circuits and Systems for Video Technology 13, 7, 560–576.

WEINBERGER, M., SEROUSSI, G., AND SAPIRO, G. 1996. LOCO-I: A low complexity, context-based, lossless image compression algorithm. In Proc. of Data Compression Conference (DCC'96). Snowbird, Utah, 140– 149.

YOON, I. AND NEUMANN, U. 2000. Web-based remote rendering with ibrac (image-based rendering acceleration and compression). Comput. Graph. Forum 19, 3, 321–330.

YOO, W., SHI, S., JEON, W. J., NAHRSTEDT, K., AND CAMPBELL, R. H. 2010. Real-time parallel remote rendering for mobile devices using graphics processing units. In ICME. IEEE, 902–907.

ZANUTTIGH, P. AND CORTELAZZO, G. 2009. Compression of depth information for 3D rendering. In Proc of 3DTV Conference: The True Vision - Capture, Transmission and Display of 3D Video. Potsdam, Germany, 1–4.