

ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΚΕΝΤΡΙΚΗΣ ΜΑΚΕΔΟΝΙΑΣ
ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΤΕ

ΑΝΑΠΤΥΞΗ ΕΡΓΑΛΕΙΟΥ ΓΙΑ ΤΗΝ ΣΧΕΔΙΑΣΗ
ΠΑΙΧΝΙΔΙΩΝ ΜΕ ΤΗΝ ΒΟΗΘΕΙΑ ΥΠΟ-ΛΟΓΙΣΤΗ

Πτυχιακή εργασία του

Γιώργου Μιχαηλίδη

Επιβλέπων: Δρ. Νικόλαος Πεταλίδης. Επιστημονικός Συνεργάτης

ΣΕΡΡΕΣ, ΦΕΒΡΟΥΑΡΙΟΣ 2016

Υπεύθυνη δήλωση

Υπεύθυνη Δήλωση: Βεβαιώνω ότι είμαι συγγραφέας αυτής της πτυχιακής εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της, είναι πλήρως αναγνωρισμένη και αναφέρεται στην πτυχιακή εργασία. Επίσης έχω αναφέρει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επίσης βεβαιώνω ότι αυτή η πτυχιακή εργασία προετοιμάστηκε από εμένα προσωπικά ειδικά για τις απαιτήσεις του προγράμματος σπουδών του Τμήματος Μηχανικών Πληροφορικής ΤΕ του Τ.Ε.Ι. Κεντρικής Μακεδονίας.

Σύνοψη

Τα πρώτα ηλεκτρονικά παιχνίδια είχαν γραφτεί εξ'ολοκλήρου σε υλισμικό. Από τότε, οι κάρτες γραφικών και οι μικροεπεξεργαστές βελτιώθηκαν, δημιουργήθηκαν κονσόλες φτιαγμένες αποκλειστικά για ηλεκτρονικά παιχνίδια, με ειδικά χειριστήρια τα οποία σου προσφέρουν διαφορετικές εμπειρίες. Η διαδικασία ανάπτυξης λογισμικού είναι ακριβή και ο σχεδιασμός γίνεται όλο πιο σύνθετος και περίπλοκος. Τα έργα γίνονται όλο πιο απαιτητικά και δαπανηρά. Δημιουργήθηκε η ανάγκη για ένα εργαλείο το οποίο να παρέχει ένα ομοιογενές περιβάλλον για την ανάπτυξη σύνθετων έργων. Ένα CASE (Computer Aided Software Engineering) tool είναι ένα λογισμικό-εργαλείο το οποίο απλοποιεί τον κύκλο ανάπτυξης ενός λογισμικού. Στο τομέα του σχεδιασμού παιχνιδιών το πιο διαδεδομένο CASE tool είναι η μηχανή γραφικών. Μια μηχανή γραφικών είναι μια σουίτα από επαναχρησιμοποιήσιμα οπτικά εργαλεία τα οποία βρίσκονται σε ένα ενιαίο περιβάλλον. Σκοπός της πτυχιακής είναι να αναγνωριστούν μοτίβα και τεχνικές δημιουργίας παιχνιδιών, ώστε να δημιουργηθεί ένα εργαλείο το οποίο να το προσεγγίζει από υψηλό επίπεδο με αφαιρέσεις για εύκολη μοντελοποίηση και αυτοματοποίηση κατά τη δημιουργία.

Περιεχόμενα

Υπεύθυνη δήλωση	2
Σύνοψη	3
Πρόλογος	9
Ευχαριστίες	10
1 Case Tools	11
1.1 Κοινές λειτουργίες	11
1.2 Γιατί;	12
1.3 Χρήση	12
1.4 Χαρακτηριστικά ενός καλού case tool	13
2 Game Engineering	14
2.1 Μηχανές Γραφικών	14
2.1.1 Ιστορία	15
2.2 Δομή μιας τυπικής ομάδας ανάπτυξης παιχνιδιών	15
2.2.1 Μηχανικοί	15
2.2.2 Artists	15
2.2.3 Game Designers	16
2.2.4 Producers	17
2.3 Τι είναι μια μηχανή γραφικών	17
2.4 Γιατί μηχανές γραφικών;	17
3 Ο πηρόνας της Μηχανής	20
3.1 Game Loop	20

3.2	ScreenSystem	20
3.3	Input System	20
4	Διαδικτύωση	24
4.1	Το πρόβλημα	24
4.1.1	Περιγραφή του προβλήματος	24
4.1.2	Κατανόηση του προβλήματος	24
4.1.3	Εξαγωγή απαιτήσεων	25
4.2	Εκπόνηση σχεδίου	26
4.2.1	Επιλογή πρωτοκόλου	26
4.2.2	Επιλογή αρχιτεκτονικής δικτύου	26
4.3	Σχεδίαση του framework	27
4.3.1	Έννοιες	27
4.3.2	Τύποι μηνυμάτων	28
4.3.3	NetworkManager	28
4.4	Υλοποίηση	29
4.4.1	Τρόπος χρήσης	29
4.4.2	Αρχιτεκτονική	29
4.4.3	Packages Module	29
4.4.4	Networking Module	31
4.4.5	API	31
4.5	Testing	36
4.6	Ανάλυση των επιδόσεων	37
4.7	Επεκτασιμότητα	38

Κατάλογος πινάκων

4.1	Network Benchmarkings	38
-----	---------------------------------	----

Κατάλογος διαγραμμάτων

2.1	Game Engine Architecture	19
3.1	game loops	21
3.2	core screensystem	22
3.3	core input	23
4.1	Network Sequence	25
4.2	Network Usage Diagram	30
4.3	Network Packages Module	32
4.4	Networking Module	33

Κατάλογος ορισμών

1	Theorem (First test theorem)	20
---	--	----

Πρόλογος

Σκοπός Σκοπός της πτυχιακής είναι να εξηγήσει θεωρητικά και πρακτικά τα κομμάτια που απαρτίζουν μια τυπική μηχανή γραφικών και πώς συνδέονται αρχιτεκτονικά μεταξύ τους, μαζί με παραδείγματα και οδηγίες για την επέκτασή τους. Για το κομμάτι του rendering, δεν έγινε απευθείας με πρωτόκολλο που επικοινωνεί με την κάρτα γραφικών, αλλά με την ανοικτού λογισμικού βιβλιοθήκη `monogame` η οποία είναι και `cross-platform`, δηλαδή ανάλογα με το περιβαλλον ανάπτυξης χρησιμοποιεί το ανάλογο πρωτόκολλο για επικοινωνία με την κάρτα γραφικών, για παράδειγμα για `Linux OS` χρησιμοποιεί `OpenGL` και για `Android OpenGL ES`, για να επικεντρωθεί στο rendering γενικά και όχι για συγκεκριμένη πλατφόρμα. Τα παραδείγματα αξιοποιούν `object-oriented` και `functional paradigms` και είναι γραμμένα σε `C#`.

Θεμελίωση Οι μηχανές γραφικών διαφέρουν με βάση τις λεπτομέρειες αρχιτεκτονικής και υλοποίησης, αλλά τα πρότυπα σχεδίασης είναι καθολικά. Πρακτικά, οι μηχανές γραφικών απαρτίζονται από τις ίδιες βασικές έννοιες. Πολλά βιβλία έχουν γραφτεί τα οποία εστιάζουν στην κάθε έννοια ξεχωριστά αλλά ελάχιστα για το πώς αυτά τα στοιχεία επικοινωνούν και αλληλεπιδρούν μεταξύ τους. Η εργασία εστιάζει στην αρχιτεκτονική των μηχανών, για το πώς οι ομάδες είναι οργανωμένες για να δουλεύουν μεταξύ τους, ποια συστήματα και μοτίβα επαναλαμβάνονται στη δημιουργία μηχανών, ποιες είναι οι απαιτήσεις για το κάθε μεγάλο υποσύστημα της μηχανής, ποια συστήματα είναι αγνωστικιστικά σε παιχνίδια ή σε είδη παιχνιδιών και ποια είναι συγκεκριμένα, και τότε σταματά η μηχανή και ξεκινά η υλοποίηση του παιχνιδιού.

Επιπρόσθετα, θα γίνει σύγκριση με άλλες δημοφιλείς μηχανές γραφικών και βιβλιοθηκών τεχνικές για `configuration management`, `versioning` και διάφορα συστήματα ανάπτυξης.

Ευχαριστίες

Ευχαριστίες (στο μπαμπά, στη μαμά, κτλ)

Κεφάλαιο 1

Case Tools

Η διαδικασία ανάπτυξης λογισμικού είναι ακριβή και ο σχεδιασμός γίνεται όλο πιο σύνθετος και περίπλοκος. Τα έργα γίνονται όλο πιο απαιτητικά και δαπανηρά. Δημιουργήθηκε η ανάγκη για ένα εργαλείο το οποίο να παρέχει ένα ομοιογενές περιβάλλον για την ανάπτυξη σύνθετων έργων. Ένα CASE (Computer Aided Software Engineering) tool είναι ένα λογισμικό-εργαλείο το οποίο απλοποιεί τον κύκλο ανάπτυξης ενός λογισμικού. Τα Case Tools γίνονται όλο και πιο δημοφιλείς, λόγω της βελτίωσης των δυνατοτήτων και της λειτουργικότητας στην ανάπτυξη της ποιότητας του λογισμικού. Η διαδικασία ανάπτυξης αυτοματοποιείται, και συντονίζεται. Το λογισμικό συντηρείται και αναλύεται εύκολα.

1.1 Κοινές λειτουργίες

- Δημιουργία ροής δεδομένων και μοντέλων οντοτήτων.
- Καθιέρωση της σχέσης μεταξύ απαιτήσεων και προτύπων.
- Ανάπτυξη του σχεδιασμού σε υψηλό επίπεδο.
- Ανάπτυξη λειτουργικών και διαδικαστικών περιγραφών
- Ανάπτυξη περιπτώσεων δοκιμών (test cases).

1.2 Γιατί;

- Γρήγορη εγκατάσταση
- Εξοικονόμηση χρόνου μειώνοντας τον χρόνο στον προγραμματισμό και στις δοκιμές.
- Οπτικοποίηση του κώδικα και της ροής δεδομένων
- Βέλτιστη χρήση των διαθέσιμων πόρων.
- Ανάλυση, ανάπτυξη και σχεδιασμό με ενιαίες μεθοδολογίες.
- Δημιουργία και τροποποίηση τεκμηρίωσης (documentation)
- Αποτελεσματική μεταφορά πληροφοριών ανάμεσα στα διάφορα εργαλεία
- Γρήγορη δημιουργία λογισμικού.

1.3 Χρήση

- Για να διευκολυνθεί η μεθοδολογία σχεδιασμού.
- Για Rapid Application Development
- Testing
- Documentation
- Project Management
- Μειωμένο κόστος συντήρησης
- Αύξηση της παραγωγικότητας:

Η Αυτοματοποίηση των διαφόρων δραστηριοτήτων των διαδικασιών ανάπτυξης και διαχείρισης του συστήματος αυξάνει την παραγωγικότητα της ομάδας ανάπτυξης.

1.4 Χαρακτηριστικά ενός καλού case tool

- Τυποποιημένη μεθοδολογία χρησιμοποιώντας τεχνικές μοντελοποίησης όπως UML.
- Flexibility: το εργαλείο πρέπει να προσφέρει τη δυνατότητα στο χρήστη να επιλέγει ποια εργαλεία να χρησιμοποιήσει.
- Strong integration: το εργαλείο πρέπει να υποστηρίζει όλα τα στάδια ανάπτυξης. Όταν γίνεται μια αλλαγή, τα στάδια τα οποία επηρεάζονται πρέπει να τροποποιούνται κατάλληλα.
- Ενσωμάτωση με εργαλεία ελέγχου.
- Reverse-engineering: δυνατότητα δημιουργίας κώδικα από δεδομένα

Κεφάλαιο 2

Game Engineering

2.1 Μηχανές Γραφικών

Στο τομέα του σχεδιασμού παιχνιδιών το πιο διαδεδομένο CASE tool είναι η μηχανή γραφικών. Μια μηχανή γραφικών είναι μια σουίτα από επαναχρησιμοποιήσιμα οπτικά εργαλεία τα οποία βρίσκονται σε ένα ενιαίο περιβάλλον. Η κεντρική λειτουργικότητα η οποία παρέχεται περιλαμβάνει τη φωτοαπόδοση σε πραγματικό χρόνο (real time rendering), τη μηχανή φυσικής και εντοπισμό συγκρούσεων (physics and collision detection), το scripting, το animation, την τεχνητή νοημοσύνη (Artificial Intelligence), τη δικτύωση (networking), τον παραλληλισμό ενεργειών (multitasking), την διαχείριση μνήμης και τον γράφο σκηνής (scene graph). Η ανάπτυξη των παιχνιδιών μέσω μιας μηχανής γραφικών γίνεται εύκολα, γρήγορα και οδηγούμενη από δεδομένα (data driven) ούτως ώστε οι δημιουργοί παιχνιδιών να μπορούν να ασχολούνται με τις λεπτομέρειες του παιχνιδιού τους. Οι μηχανές αναπτύσσονται από ομάδες που απαρτίζονται όχι μόνο από προγραμματιστές, αλλά και από μαθηματικούς, φυσικούς κλπ. Η κάθε υπο-ομάδα εστιάζει σε ένα συγκεκριμένο κομμάτι, όπως οι φυσικοί με τον εντοπισμό συγκρούσεων, και αρχιτέκτονες λογισμικού σχεδιάζουν το πώς τα κομμάτια συνδέονται και αλληλεπιδρούν μεταξύ τους, χωρίς να τους απασχολούν οι λεπτομέρειες σχεδίασης του κάθε κομματιού.

2.1.1 Ιστορία

Η ιστορία των Βιντεοπαιχνιδιών, αρχίζει στα τέλη της δεκαετίας του '40. Προς τα τέλη του '50 και στα μέσα του '60, στην Αμερική, αρχίζουν να μπαίνουν στην καθημερινή μας ζωή, οι υπολογιστές. Για την ακρίβεια, οι κεντρικοί υπολογιστές. Από εκείνη την περίοδο, τα βιντεοπαιχνίδια έκαναν την εμφάνιση τους, στις κονσόλες, στα φλίπερ, στους υπολογιστές, αλλά και στις φορητές κονσόλες. Από τότε η δημιουργία παιχνιδιών έχει γιγαντιωθεί έχοντας ένα τεράστιο κομμάτι της παγκόσμιας οικονομίας. Πλέον ο ανταγωνισμός είναι τεράστιος, τα βιντεοπαιχνίδια κυκλοφορούν για διάφορες κονσόλες με πολύ απαιτητικά γραφικά και με πολύ γρήγορο ρυθμό.

2.2 Δομή μιας τυπικής ομάδας ανάπτυξης παιχνιδιών

Πριν από την ανάλυση της δομής της μηχανής, θα γίνει ανάλυση της δομής ομάδας η οποία θα την χρησιμοποιεί για να αναπτυχθούν στοχευμένα εργαλεία για το κάθε πρόβλημα της κάθε υπο-ομάδας.

2.2.1 Μηχανικοί

Οι μηχανικοί σχεδιάζουν και υλοποιούν το λογισμικό του παιχνιδιού και τα εργαλεία τα οποία χρησιμοποιούνται για την ανάπτυξή του. Οι δύο μεγάλες κατηγορίες μηχανικών είναι οι

- runtime programmers οι οποίοι ασχολούνται με τη μηχανή και το παιχνίδι
- tool programmers οι οποίοι γράφουν tools τα οποία αυτοματοποιούν και ευκολύνουν την διαδικασία ανάπτυξης.

Οι μηχανικοί έχουν είτε κάποια ειδικότητα, για παράδειγμα ειδικότητα στη τεχνητή νοημοσύνη, είναι generalists, δηλαδή κατέχουν από όλα τα στοιχεία και μπορούν να λύσουν προβλήματα που κατά τη διάρκεια ανάπτυξης.

2.2.2 Artists

Οι artists παράγουν όλο το οπτικοακουστικό κομμάτι του παιχνιδιού, το οποίο είναι βασικό κομμάτι για το χαρακτήρα του παιχνιδιού. Χωρίζονται στις εξής κατηγορίες

- Concept artists οι οποίοι σχεδιάζουν σκίτσα και πίνακες τα οποία παρέχουν στην ομάδα την εικόνα του τελικού παιχνιδιού. Παρέχουν οπτική καθοδήγηση στην ομάδα καθ' όλη τη διάρκεια του κύκλου ανάπτυξης.
- 3D Modelers οι οποίοι είναι υπεύθυνοι για την τρισδιάστατη γεωμετρία του εικονικού κόσμου του παιχνιδιού. Απαρτίζονται από τους foreground modelers οι οποίοι σχεδιάζουν χαρακτήρες, οχήματα, οπλα και αντικείμενα του τρισδιάστατου κόσμου background modelers οι οποίοι σχεδιάζουν την στατικό περιβάλλον πχ κτήρια
- Texture artists οι οποίοι σχεδιάζουν τις δισδιάστατες εικόνες που καλύπτουν τα τρισδιάστατα μοντέλα
- Lighting artists που ορίζουν τις στατικές και δυναμικές πηγές φωτός και δουλεύουν με το χρώμα, την κατεύθυνση και την κατεύθυνση του φωτός.
- Animators οι οποίοι σχεδιάζουν την κίνηση των χαρακτήρων και των αντικειμένων
- Motion capture actors οι οποίοι παρέχουν ακατέργαστα δεδομένα κίνησης για να επεξεργαστούν οι animators και να τα ενσωματώσουν στο παιχνίδι.
- Sound designers οι οποίοι παράγουν τα εφέ και τη μουσική.
- Voice actors τους οποίους η φωνή ηχογραφείται και χρησιμοποιείται για τους χαρακτήρες στο παιχνίδι

2.2.3 Game Designers

Η δουλειά ενός game designer είναι να σχεδιάσει το διαδραστικό τμήμα του παιχνιδιού, το gameplay. Ασχολούνται με τον σχεδιασμό επιπέδων, την ιστορία της αλληλεπιδράσεις μεταξύ των χαρακτήρων στο παιχνίδι με τους στόχους, σκοπούς και κανόνες του παιχνιδιού. Σχεδιάζουν το κάθε επίπεδο μονδικά και αποφασίζουν για τη γεωμετρία στο περιβάλλον, πότε και που εμφανίζονται χαρακτήρες και διάφορα αντικείμενα, πως γίνονται οι μεταβάσεις μεταξύ διάφορων σκηνών κλπ.

2.2.4 Producers

Ο ρόλος του producer διαφέρει από στούντιο σε στούντιο. Η βασική του δουλειά είναι να προγραμματίζει και να δρομολογεί τις διάφορες εργασίες και να λειτουργεί ως ο συνδετικός κρίκος μεταξύ των ατόμων που παίρνουν ηγετικές αποφάσεις και την ομάδα ανάπτυξης. Οι producers είναι χαρακτηριστικό των AAA εταιριών, όπου υπάρχουν πολλά τμήματα και πολλοί εργαζόμενοι.

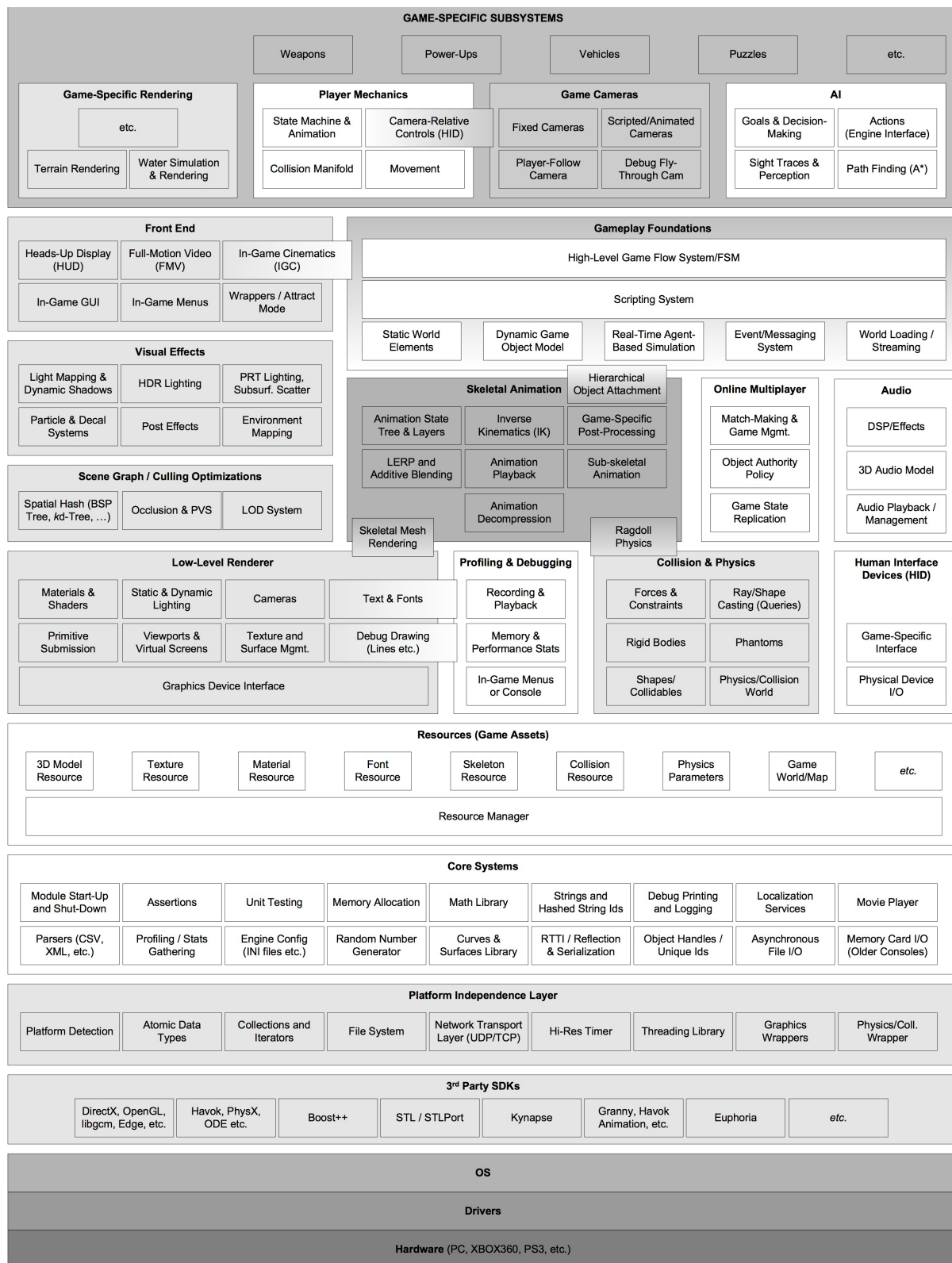
2.3 Τι είναι μια μηχανή γραφικών

Η πρώτη αναφορά σε μηχανή γραφικών έγινε στα μέσα της δεκαετίας του 90 και αναφερόταν στο δημοφιλές παιχνίδι Doom του οποίου η αρχιτεκτονική διαχώριζε τα βασικά συστήματα του παιχνιδιού, όπως rendering system, collision detection system, audio system, asset system κλπ. Η αξία αυτού του διαχωρισμού εκτιμήθηκε από την κοινότητα όταν οι προγραμματιστές ξεκίνησαν να πουλάνε άδειες για το λογισμικό, επαναχρησιμοποιούσαν εργαλεία προηγούμενων παιχνιδιών με δημιουργία νέων assets. Μικρότερα στούντιο τροποποιούσαν εκδόσεις υπάρχων παιχνιδιών χρησιμοποιώντας το SDK. Πολλά παιχνίδια γράφτηκαν με σκοπό να επαναχρησιμοποιηθούν κομμάτια κώδικα και modding. Πολλές μηχανές όπως η μηχανή του Quake III γράφτηκαν με τρόπο ώστε να είναι εύκολα προσαρμόσιμες χρησιμοποιώντας scripting, με σκοπό την εμπορευματοποίηση μέσω licensing. Η διαχωριστική γραμμή μεταξύ του παιχνιδιού και της μηχανής δεν μπορεί να οριστεί με ακρίβεια. Πολλές μηχανές μπορεί να περιέχουν συγκεκριμένα μέρη που αφορούν συγκεκριμένη λειτουργία του παιχνιδιού. Η μεγάλη διαφορά είναι στο data-driven architecture όπου οι κανόνες και τα στοιχεία δεν είναι hard-coded αλλά διαβάζονται από εξωτερικό αρχείο. Οι μηχανές έχουν τα όριά τους ανάλογα με τα είδη παιχνιδιού στα οποία η μηχανή εστιάζει, σε ποιες πλατφόρμες, σε τι στιλ γραφικών, σε ποια αρχιτεκτονική της gpu κλπ.

2.4 Γιατί μηχανές γραφικών;

Η αφαίρεση πάντα βοηθούσε τον εγκέφαλο να λειτουργήσει καλύτερα και να κατανοήσει αλληλεπιδράσεις μεταξύ συστημάτων και περίπλοκες έννοιες. Οι μηχανές γραφικών απαλλάζουν τους γραφίστες και τους προγραμματιστές από τις τεχνικές λε-

πτομέριες, και εστιάζουν στην αισθητική και στο gameplay. Επίσης με την αποσύνδεση των συστημάτων έχουμε πιο προβλέψιμη συμπεριφορά, επεκτασιμότητα των υποσυστημάτων ως υποσυστήματα και ευκολη δοκιμαστικότητα.



Διάγραμμα 2.1: Game Engine Architecture

Κεφάλαιο 3

Ο πηρύνας της Μηχανής

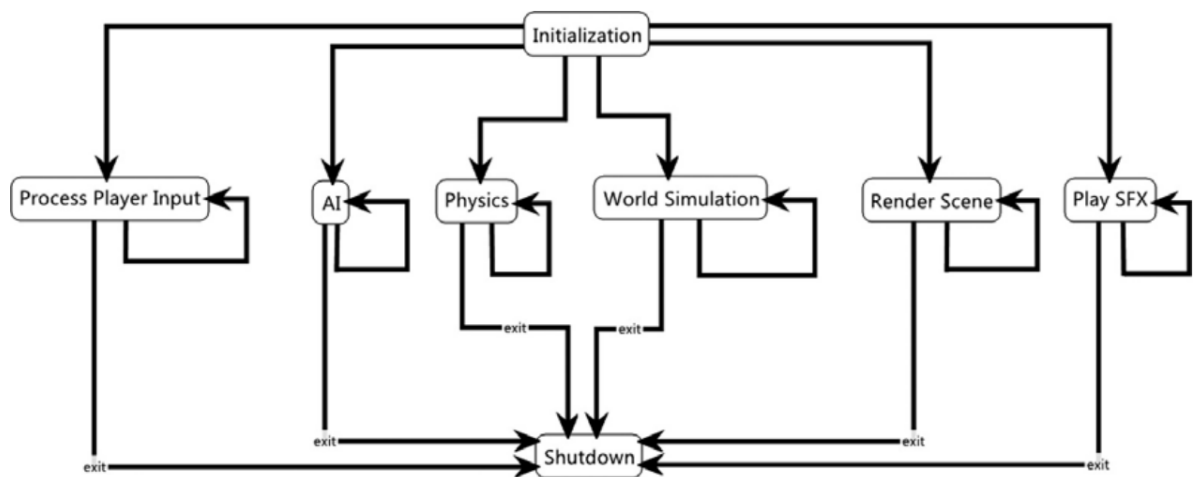
Όταν ένα framework επεκτείνεται και διακλαδώνεται και σε πολλές υπο-βιβλιοθήκες, στηρίζεται στη θεμελίωση ενός framework core, του πηρύνα της βιβλιοθήκης, στον οποίο βασίζονται οι διάφορες επεκτάσεις και υλοποιήσεις. Το API του πηρύνα πρέπει να αποτελείται από αφαιρέσεις, οι οποίες εκθέτουν μόνο τα απολύτως απαραίτητα, ώστε οι υποβιβλιοθήκες που χρησιμοποιούν τον πηρύνα να μην δεσμεύονται σε υλοποιήσεις και να περιορίζονται σε συγκεκριμένο pipeline χρήσης για αποφυγή απρόβλεπτων συμπεριφορών. Tulach 2008

Theorem 1 (First test theorem). This is a test theorem.

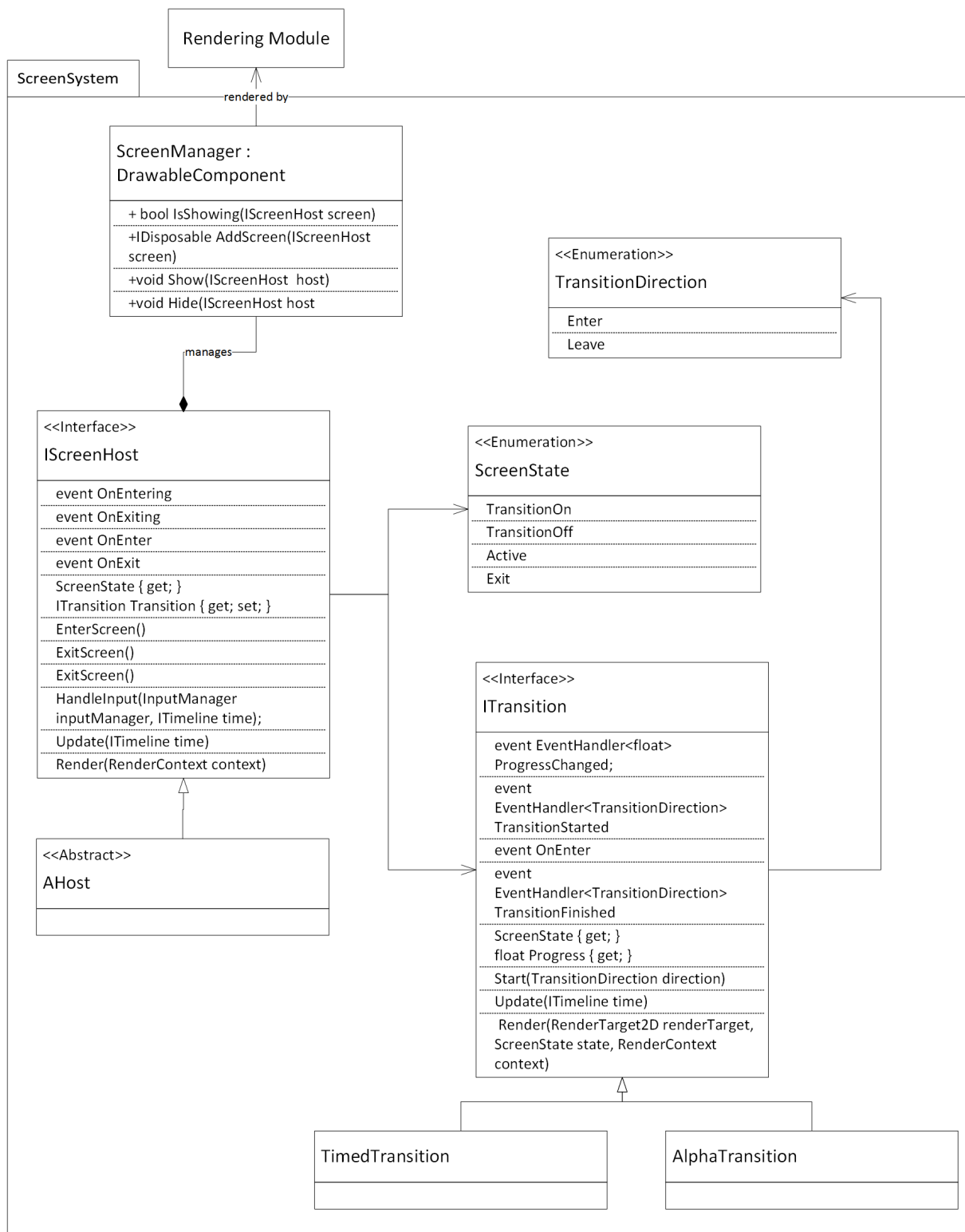
3.1 Game Loop

3.2 ScreenSystem

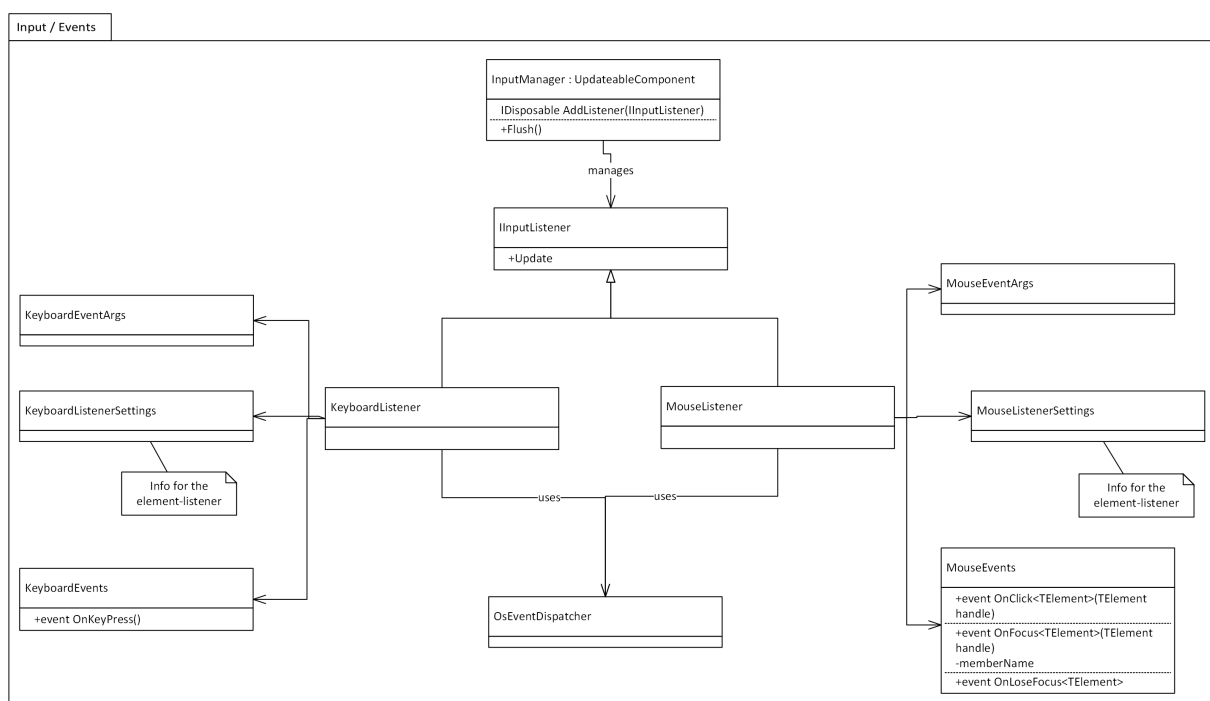
3.3 Input System



Διάγραμμα 3.1: game loops



Διάγραμμα 3.2: core screensystem



Διάγραμμα 3.3: core input

Κεφάλαιο 4

Διαδικτύωση

Διαδικτύωση στα ηλεκτρονικά παιχνίδια έχουμε όταν περισσότεροι από ένας παίκτες σε διαφορετικές πλατφόρμες ή υπολογιστές, μοιράζονται και αλληλεπιδρούν στο ίδιο εικονικό περιβάλλον.

4.1 Το πρόβλημα

4.1.1 Περιγραφή του προβλήματος

Διάφοροι παίκτες σε διάφορα σημεία του πλανήτη θέλουν να μοιραστούν ένα εικονικό περιβάλλον σε πραγματικό χρόνο με σκοπό την συνεργασία ή την αντιπαλότητα. Ο κόσμος είναι ένα υπερσύνολο του offline κόσμου με επιπλέον στοιχεία κοινωνικοποίησης όπως η επικοινωνία μέσω μηνυμάτων ή φωνής.

4.1.2 Κατανόηση του προβλήματος

Ένας εικονικός κόσμος, περιλαμβάνει πολλές οντότητες οι οποίες αλληλεπιδρούν μεταξύ τους μέσω των μηχανισμών, νόμων και κανόνων που διέπουν τον κόσμο. Παράδειγμα μηχανισμού είναι η προσομοίωση του φυσικού κόσμου, όπου οι οντότητες αναποκρίνονται σε νόμους της φυσικής.

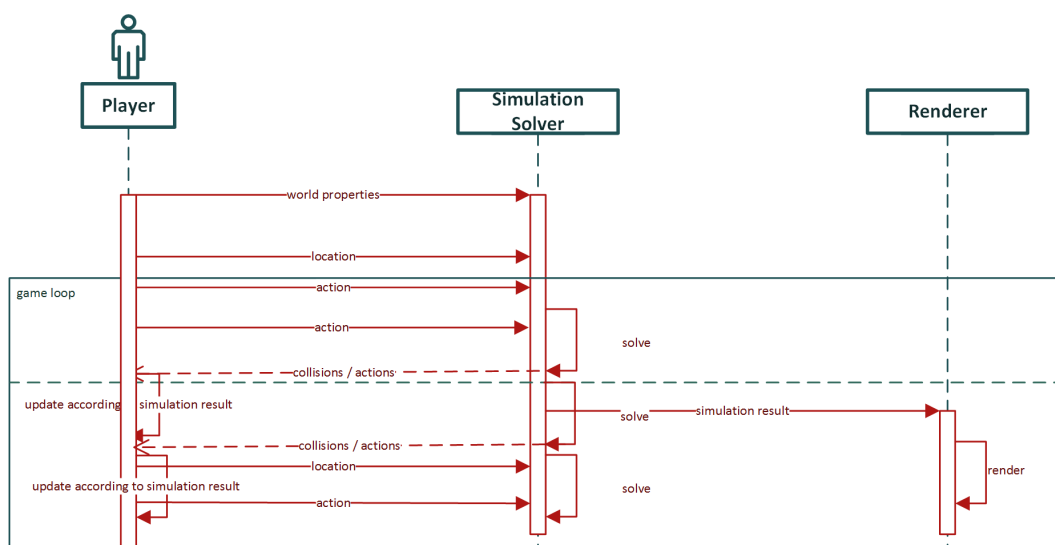
Κατά την ενημέρωση του κόσμου, ο προσομοιωτής χρησιμοποιώντας τους νόμους, τους κανόνες και τους μηχανισμούς που διέπουν τον κόσμο, παίρνει ως είσοδο τις οντότητες, τα ειδικά βάρη των ιδιοτήτων τους, την απόλυτη θέση τους στο σύστημα συντεταγμένων του κόσμου, και το χρονικό διάστημα της προσομοίωσης και αναλύει την

προσομοίωση. Η ανάλυση της προσομοίωσης για να είναι επιτρεπτά ακριβής πρέπει να γίνεται περίπου 80-100 φορές το δευτερόλεπτο. [αναφορά σε πηγή]

Στο τέλος της προσομοίωσης, η μηχανή γραφικών αποτυπώνει τον κόσμο στις εξόδους με αναφορές σε στατικά assets, σε αλγόριθμους παραγωγής δυναμικών assets για την αναπαράσταση του κόσμου.

4.1.3 Εξαγωγή απαιτήσεων

Με βάση το πρόβλημα καταλήγουμε στο παρακάτω διάγραμμα ακολουθίας.



Διάγραμμα 4.1: Network Sequence

Βλέποντας το διάγραμμα καταλαβαίνουμε ότι σε η διαδικασία rendering περιλαμβάνει στατικά στοιχεία και αλγόριθμους, τα οποίοι μπορούν να φορτώνονται τοπικά στον κάθε υπολογιστή, και δεν είναι απαραίτητα για την επίλυση της προσομοίωσης. Τα απαραίτητα στοιχεία για την προσομοίωση τα οποία πρέπει να μοιράζονται μεταξύ των παιχτών είναι:

- Οι ιδιότητες της οντότητας με βάση τους νόμους του κόσμου. Οι ιδιότητες αυτές δεν ενημερώνονται συχνά.
- Οι αλλαγές στο σύστημα συντεταγμένων και οι διάφορες ενέργειες της κατευθυνόμενης οντότητας κατά την πάροδο του χρόνου. Οι αλλαγές τοποθεσίας και ενέργειες γίνονται πολλές φορές ανά δευτερόλεπτο. Η προσομοίωση για να είναι ακριβής πρέπει να ενημερώνεται για τις διάφορες ενέργειες σε πραγματικό

χρόνο.

4.2 Εκπόνηση σχεδίου

Η ανάγκη αποστολής πολλών μηνυμάτων ανά δευτερόλεπτο οδηγεί στη χρήση των network sockets. Τα sockets χρησιμοποιώντας ένα IP και Port και επιτρέπουν την αποστολή και παραλαβή μηνυμάτων με βάση κάποιου πρωτόκολλου.

4.2.1 Επιλογή πρωτοκόλλου

- TCP (transmission control protocol), είναι το πιο συχνά χρησιμοποιημένο πρωτόκολλο. Η διασύνδεση με TCP είναι αξιόπιστη και τα μηνύματα παραλαμβάνονται στη σειρά αποστολής. Η αξιοπιστία όμως έρχεται με ένα μικρό κόστος απόδοσης.
- UDP To UDP (user diagram protocol) δεν περιλαμβάνει την αξιοπιστία και την εγγύηση της αλληλουχίας μηνυμάτων. Η απουσία λειτουργιών όμως, το κάνει το πιο γρήγορο σε αποστολή μηνυμάτων πρωτόκολλο.

Η επιλογή πρωτοκόλλου γίνεται ανάλογα με το γενικότερο πλαίσιο και τη συγκεκριμένη χρήση του πακέτου αποστολής. Στην αποστολή της τοποθεσίας, το οποίο γίνεται 20φορές / δευτερόλεπτο, η αξιοπιστία δεν είναι το βασικότερο, αλλά η απόδοση. Στην αποστολή των στοιχείων του χρήστη κατά την έναρξη, ή ενός γραπτού μηνύματος πρέπει να είναι αξιόπιστη.

4.2.2 Επιλογή αρχιτεκτονικής δικτύου

Οι αρχιτεκτονικές δικτύου χωρίζονται ανάλογα με το που γίνεται η επίλυση και επεξεργασία της προσομοίωσης.

- Client-server model: στο οποίο ο client απλά κάνει render και το μεγαλύτερο κομμάτι της λογικής και της προσομοίωσης τρέχει στον server. Ο server στέλνει οδηγίες στον client για το τι να κάνει render και ο client απλά υπακούει.
- Client on top of server model: ο client είναι και server, δηλαδή οι μηχανές που έχουν τον client έχουν και τον server.

- Peer-to-peer: οι μηχανές συμπεριφέρονται μερικώς ως clients και μερικώς ως servers, δηλαδή έχουν και στοιχεία λογικής και επεξεργασίας.

Η client-server αρχιτεκτονική εφαρμόζεται σε παιχνίδια τα οποία περιλαμβάνουν πολύ μεγάλους κόσμους και η προσομοίωση περιλαμβάνει αλληλεπιδράσεις μεταξύ πολλών οντοτήτων. Οι προσομοιώσεις αυτές γίνονται σε εξειδικευμένες μηχανές και όχι στον προσωπικό υπολογιστή του κάθε χρήστη γιατί χρειάζονται μεγάλους υπολογιστικούς πόρους. Επίσης ο server μπορεί να λύσει race conditions και να λειτουργήσει ως "διαιτητής" μεταξύ δύο οντοτήτων οι οποίες ζητούν πρόσβαση στο ίδιο σύστημα κατά το ίδιο χρονικό διάστημα. Οι αρχιτεκτονικές στις οποίες ο client περιλαμβάνει στοιχεία επεξεργασίας του κοινόχρηστου κόσμου, είναι πιο δύσκολες στην ανάπτυξη συντήρηση γιατί δημιουργούνται race conditions στο χρονικό διάστημα αποστολής-παραλαβής μηνυμάτων που προορίζονται σε αλληλοεξαρτώμενες λειτουργίες.

4.3 Σχεδίαση του framework

Κατά το σχεδιασμό διαδικτυακών παιχνιδιών πολλά μοτίβα και στερεότυπα κώδικα επαναλαμβάνονται χωρίς να συμβάλλουν στην λογική ή μηχανισμούς. Σκοπός του framework είναι να μειώσει και να απλοποιήσει τον επαναλαμβανόμενο κώδικα και να προμηθεύσει τον χρήστη με μοτίβα και μοντέλα ούτως ώστε να εστιάσει μόνο στα απαραίτητα.

4.3.1 Έννοιες

Serialization Σε μια αντικειμενοστραφή γλώσσα χρησιμοποιούνται κλάσεις για να μοντελοποιήσουν το πρόβλημα. Όμως τα αντικείμενα των κλάσεων δεν μπορούν να σταλούν μέσω network socket στην αρχική τους μορφή, πρέπει να γίνει μια μετατροπή σε binary για να είναι συνεπής με το format των δεδομένων που αποστέλλονται μέσω του socket. Κατά την αποστολή έχουμε το serialization των αντικείμενων, και κατά την παραλαβή το deserialization του πακέτου στο αντικείμενο που αντιπροσωπεύει.

4.3.2 Τύποι μηνυμάτων

Οι διάφοροι τύποι μηνυμάτων χρησιμοποιούνται για να ξεχωρίσουν την κατάσταση στην οποία βρίσκεται ο client ή ο server.

- Αλλαγή κατάστασης
 - Connecting
 - Connected
 - Disconnecting
 - Disconnected

- Data
- ErrorMessage
- WarningMessage
- VerboseMessage
- ConnectionApproval
- DiscoveryRequest
- DiscoveryResponse

Οι τύποι μηνυμάτων μπορούν να μοντελοποιηθούν ως ένα enum και να προστεθεί η πληροφορία τους στο πρώτο byte του serialized μηνύματος. Το πρώτο byte του παραληφθέντα μηνύματος περιλαμβάνει τον τύπο του μηνύματος.

4.3.3 NetworkManager

Ο network manager είναι υπεύθυνος για την διαχείριση των sockets, την αποστολή / παραλαβή μηνυμάτων, ενθυλακώνει λειτουργίες για την διαδίκτυωση και είναι ο πηρύνας του framework και επεκτείνεται από τον client, server οι οποίοι προσθέτουν λειτουργίες.

4.4 Υλοποίηση

4.4.1 Τρόπος χρήσης

Η διαδικασία χρήσης πρέπει να είναι εξορθολογιστική και ξεκάθαρη για εύκολη και γρήγορη ανάπτυξη και περιοριστική για αποφυγή bugs και προβλημάτων.

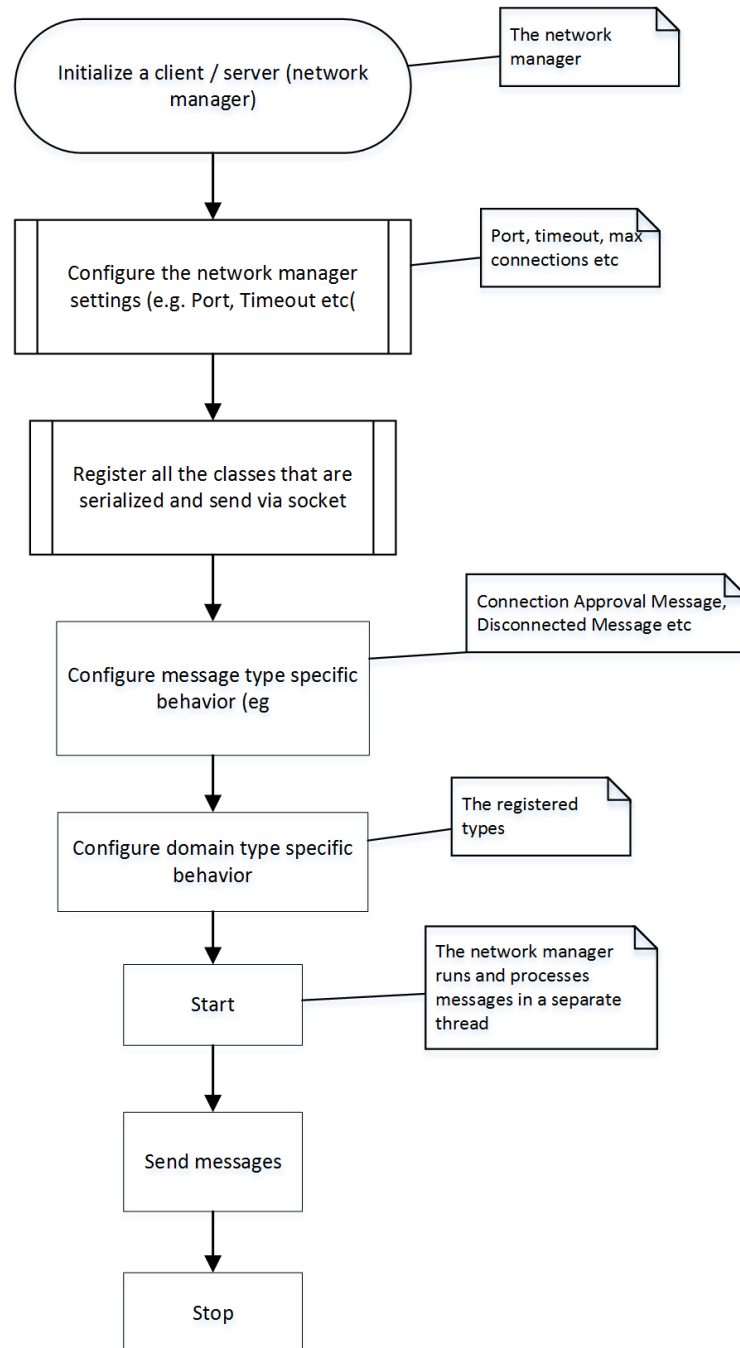
4.4.2 Αρχιτεκτονική

Η αρχιτεκτονική στηρίζεται στην ανταλλαγή μηνυμάτων-κλάσεων και χωρίζεται στα παρακάτω επίπεδα.

- **Packages** Χειρίζεται το serialization, αναλύει τα εισερχόμενα μηνύματα και χειρίζεται τις ανακατευθύνσεις του εκτελούμενου κώδικα κατά την παραλαβή μηνυμάτων.
- **Networking** Διαχειρίζεται τις συνδέσεις των χρηστών, τα sockets και ο,τι έχει να κάνει με διασύνδεση.
- **Auditing** Αφαιρετικό επίπεδο για logging. Ο χρήστης μπορεί να ενσωματώσει τη δική του λογική παρακολούθησης μηνυμάτων.
- **Infrastructure** Το επίπεδο αυτό περιέχει επαναχρησιμοποιήσιμο κώδικα των πακέτων όπως το ConcurrentRepository, για thread-safe αποθήκευση κλειδιών και τιμών, και το ParallelBufferBlock το οποίο εκτελεί κώδικα σε buffer άλλου thread για ασύγχρονη επεξεργασία.

4.4.3 Packages Module

Το serialization των μηνυμάτων πρέπει να είναι ντερερμενιστικό ώστε να αναγνωρίζεται ο τύπος του μηνύματος και η κλάση την οποία αντιπροσωπεύει και ελαφρύς ώστε να μην επιβαρύνεται το network με επιπλέον δεδομένα. Ο ενσωματωμένος serializer χρησιμοποιεί reflection για να αναγνωρίσει τα primitive properties της κλάσης. Για προχωρημένο serialization περίπλοκων τύπων, ο χρήστης έχει τη δυνατότητα να συμπεριλάβει τον δικό του serializer με το implementation του IPackageSerializer interface και την χωρήγηση του σημειώνοντας την κλάση με το PackageSerializerAttribute και τον τύπο του serializer.



Διάγραμμα 4.2: Network Usage Diagram

Κατά την προετοιμασία του network manager, ο χρήστης καλείται να καταχωρίσει στο Container ποιες κλάσεις θα χρησιμοποιηθούν κατά την ανταλλαγή μηνυμάτων. Οι κλάσεις που προωρίζονται για serialization σημειώνονται με κάποιο Attribute και η καταχώριση μπορεί να γίνει δυναμικά με reflection. Όταν τελειώσει η καταχώριση, το container χρησιμοποιώντας ένα ντετερμινιστικό αλγόριθμο ταξινομώντας με βάση το όνομα της κλάσης στο χώρο ονομάτων κτίζει ένα thread safe repository στο οποίο καταχωρείται ένα μοναδικό byte για αναγνωριστικό του κάθε τύπου και πληροφορίες για την χρήση του τύπου.

Στο τέλος της καταχώρισης και της κατασκευής αλγόριθμου αντιστοίχισης αντικειμένων και λογικής, ο χρήστης μπορεί να καταχωρίσει εκτελέσιμο κώδικα υπό μορφή function ο οποίος εκτελείται ασύγχρονα κατά την παραλαβή κάποιου μηνύματος-κλάσης, τύπου μηνύματος, ή συμβάντος συγκεκριμένης σύνδεσης.

4.4.4 Networking Module

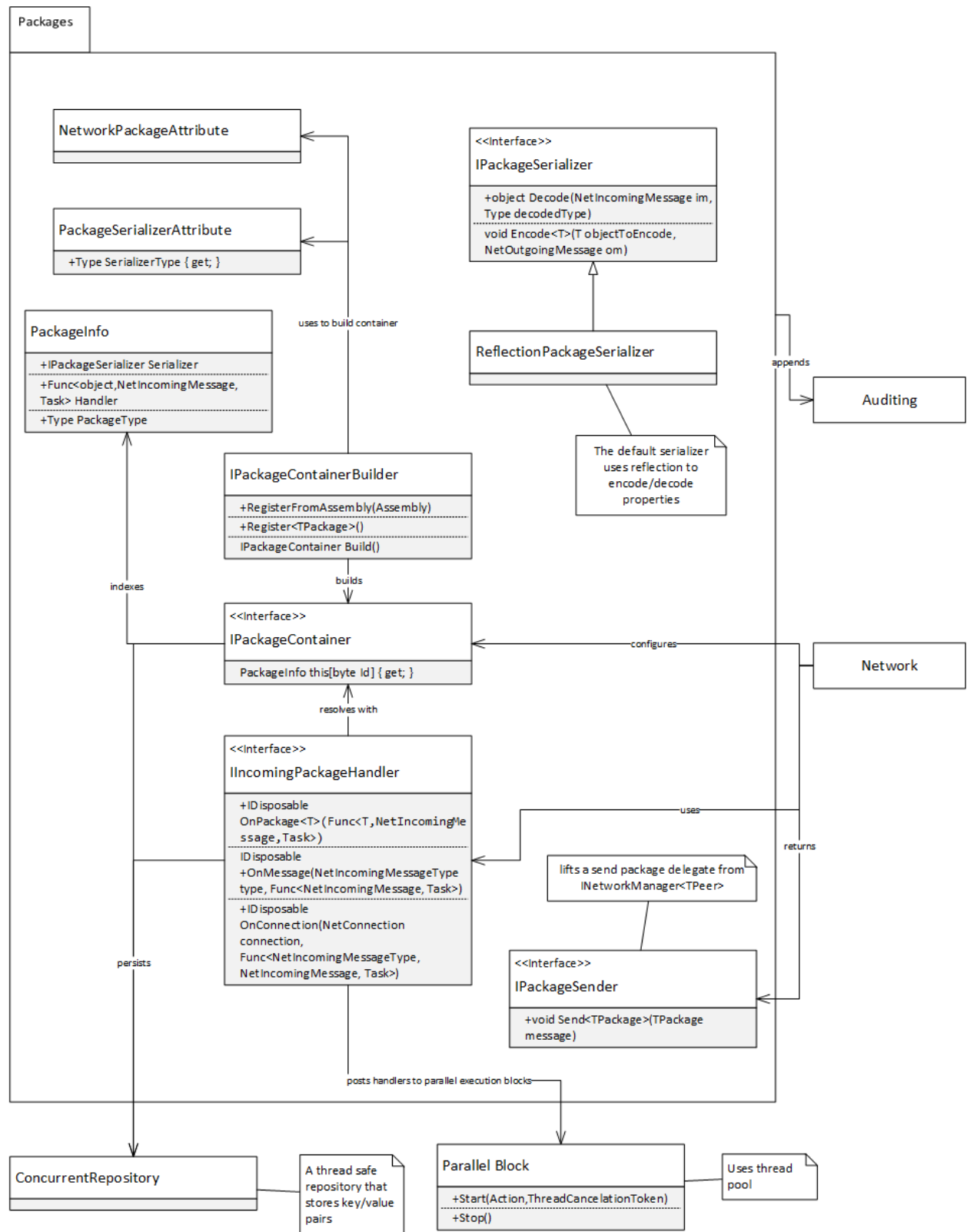
Οι ρυθμίσεις του network manager κτίζονται από αφαιρέσεις. Η χρήση πρωτοκόλλων και τεχνικών γίνεται με άγνοια της υλοποίησης και λεπτομερειών. Η σχεδίαση επιτρέπει την αποστολή μηνύματος ανεξαρτήτου πρωτοκόλου, τρόπο αποστολής και παραλήπτη. Ο network manager περιλαμβάνει τεχνικές function lifting για αποστολή μηνυμάτων. Ο χρήστης μπορεί να ρυθμίσει διάφορους τρόπους αποστολής ανάλογα με το περιβάλλον χρήση, και να τους κρύψει πίσω από functions.

4.4.5 API

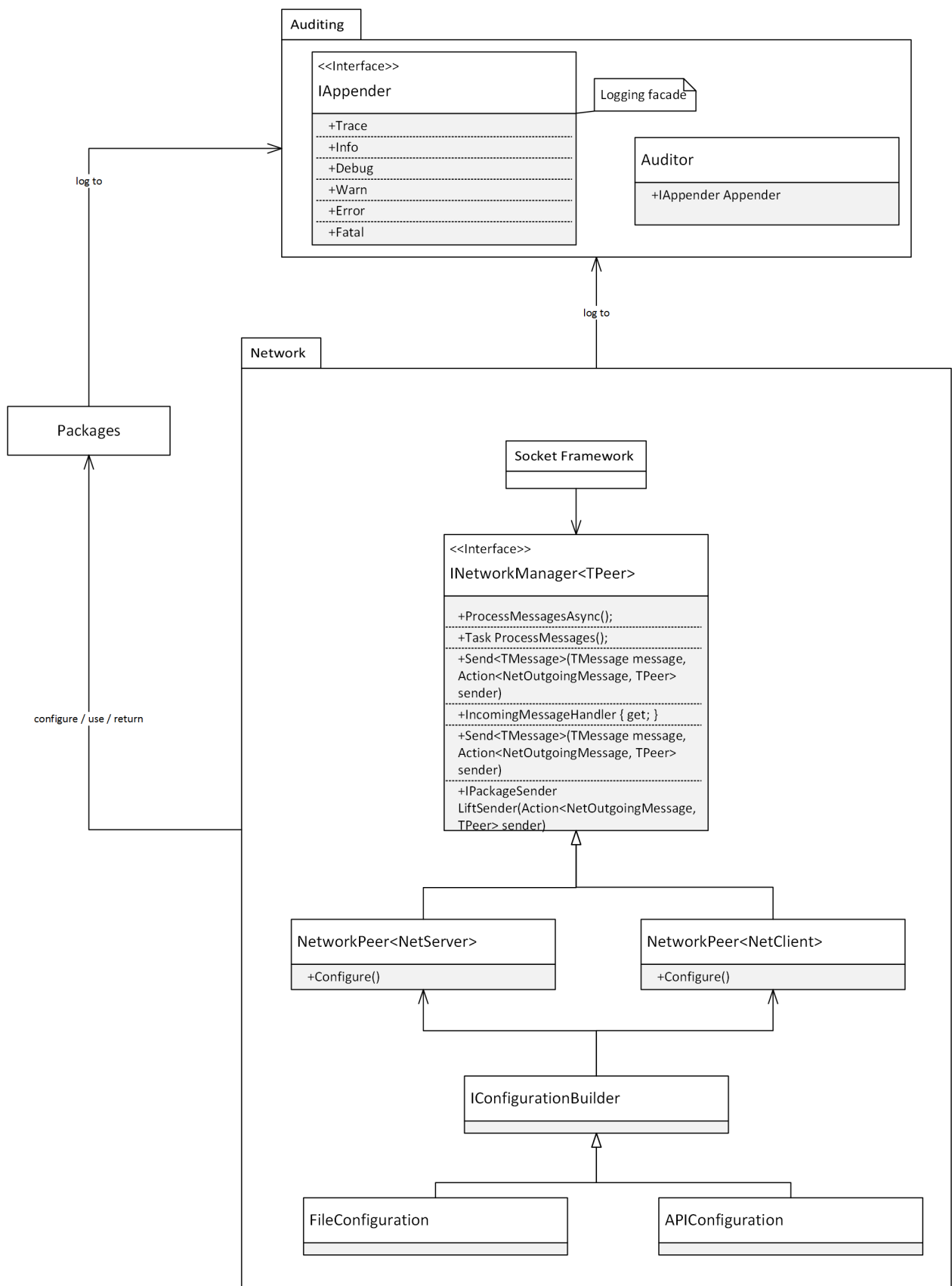
Συνοψη επισκόπηση του API.

- Κλάσεις που προορίζονται για ανταλλαγή μηνυμάτων

```
[ GinetPackage ]
    // optional
[ PackageSerializer ( typeof( MyCustomSerializer ) ) ]
public class MyPackage
{
    public string Message { get; set; }
}
```



Διάγραμμα 4.3: Network Packages Module



Διάγραμμα 4.4: Networking Module

- Πύθμηση server

```
var server = new NetworkServer("MyServer",
builder =>
{
    // via reflection
    builder.RegisterPackages(
        Assembly.Load("Packages.Assembly.Name"));
    // or manually
    builder.RegisterPackage<MyPackage>();
}
cfg =>
{
    cfg.Port = 1234;
    cfg.ConnectionTimeout = 5.0f;
    cfg.MaxConnections = 10;
    // Additional configuration
}
// optional, logging output, default is the one supplied
out: new ActionAppender(Console.WriteLine)
```

- Καταγραφή κινήσεων

```
server.IncomingMessageHandler.LogTraffic();
```

- Απάντηση κατά την παραλαβή μηνύματος

```
server.Broadcast<ChatMessage>((msg, im, om) =>
{
    server.Out.Info($"Received_{msg.Message}");
    server.SendToAllExcept(om, im.SenderConnection,
        NetDeliveryMethod.ReliableOrdered, channel: 0);
},
packageTransformer: msg =>
msg.Message += "this_is_broadcasted");
```

```
server.BroadcastExceptSender<ChatMessage>((sender, msg) =>
{
    server.Out.Info($"Broadcasting_{msg.Message}._
        Received_from:{sender}");
});
```

- Κώδικας ο οποίος εκτελείται κατά την παραλαβή συγκεκριμένου τύπου μηνύματος

```
server.IncomingMessageHandler.OnMessage(
    NetIncomingMessageType.ConnectionApproval,
    incomingMsg =>
{
    // Configure connection approval
    var parsedMsg = server
        .ReadAs<ConnectionApprovalMessage>(
            incomingMsg);
    if (parsedMsg.Password ==
        "my_secret_and_encrypted_password")
    {
        incomingMsg.SenderConnection.Approve();
        incomingMsg.SenderConnection.Tag =
            parsedMsg.Sender;
    }
    else
    {
        incomingMsg.SenderConnection.Deny();
    }
});
```

- Εκτελέσιμος κώδικας κατά την παραλαβή συγκεκριμένου μηνύματος-κλάσης

```
server.IncomingMessageHandler
    .OnPackage<MyPackage>((msg, sender) =>
        Console.WriteLine(
            $" {msg.Message} _from_{sender.SenderConnection}"));
```

- Αποστολή μηνύματος-κλάσης

```
server.Send(
    new MyPackage { Message = "Hello" },
    (outgoingMessage, peer) =>
        peer.SendMessage(outgoingMessage,
            NetDeliveryMethod.ReliableOrdered));
```

- Lift αποστολής μηνύματος

```
var packageSender = client.LiftSender((msg, peer) =>
    peer.SendMessage(msg,
        NetDeliveryMethod.ReliableOrdered));

packageSender.Send(new MyPackage { Message = "Hello" });
```

- Διαγραφή handler

```
var handlerDisposable = server.IncomingMessageHandler
    .OnPackage<MyPackage>((msg, sender) => {}));

handlerDisposable.Dispose();
```

- Η ρύθμιση και το API του client είναι πανομοιότυπο με του server. Στη θέση του *NetworkServer* χρησιμοποιείται ο *NetworkClient* όπου και τα δύο υλοποιούν τον *NetworkManager<TPeer>* where *TPeer:NetPeer*

4.5 Testing

Μια καλή αρχιτεκτονική υποστηρίζεται από την ευκολία της δοκιμαστικότητας ανά μονάδα (unit testability). Το framework χρησιμοποιεί functional τεχνικές, οι οποίες εύκολα μπορούν να αντικατασταθούν με mocks. Ο χρήστης μπορεί να απομιμήσει την παραλαβή ή αποστολή πακέτων και να ενημερώνει την μονάδα επεξεργασίας μηνυμάτων από το τρέχων thread.

[TestMethod]

```

public void Incoming_Package_GetsHandled()
{
    // Arrange

    var client = new NetworkClient(
        builder =>
            builder.Register<MyPackage>(),
        cfg=> {});

    var mockedHandler =
        new Mock<Func<MyPackage, NetIncomingMessage>>()
        ;
    mockedHandler.Setup( x =>
        x( It.IsAny<MyPackage>() ) );

    client
        .IncomingMessageHandler
        .OnPackage<MyPackage>(mockedHandler);

    // Act

    client.IncomingMessageHandler.SimulateReceive(
        new MyPackage());
    client.ProcessMessages().RunSynchronously();

    // Assert

    mockedHandler.Verify(x=>x(), Times.Once());
}

```

Επίσης παρέχεται η δυνατότητα της προσομοίωσης χαμένων πακέτων, για να δοκιμαστεί πως η εφαρμογή ανταποκρίνεται σε περιβαλλον κακής διαδικτύωσης και να αναπτυχθούν τεχνικές network prediction για εξομάλυνση της κίνησης.

4.6 Ανάλυση των επιδόσεων

Benchmarkings here

Πίνακας 4.1: Network Benchmarkings

Κίνητρα	Παραδείγματα ευρημάτων	Αριθμός μελετών
Ταύτιση με το έργο	Ξεκάθαροι στόχοι	20
Καλό management	Ομαδικότητα	16
Συμμετοχή υπαλλήλων	Συμμετοχή στις αποφάσεις	16
Προοπτικές εξέλιξης	Προοπτικές προαγωγής	15
Ποικιλία στην εργασία	Καλή χρήση ικανοτήτων	14
Αίσθηση του να ανήκεις κάπου	Υποστηρικτικές σχέσεις	14
Αμοιβές και κίνητρα	Αυξημένος μισθός	14

4.7 Επεκτασιμότητα

Lobbies and matchmakings

Γλωσσάρι

API Η Διεπαφή Προγραμματισμού Εφαρμογών (αγγλ. API, από το Application Programming Interface), γνωστή και ως Διασύνδεση Προγραμματισμού Εφαρμογών (για συντομία διεπαφή ή διασύνδεση), είναι η διεπαφή των προγραμματιστικών διαδικασιών που παρέχει ένα λειτουργικό σύστημα, βιβλιοθήκη ή εφαρμογή προκειμένου να επιτρέπει να γίνονται προς αυτά αιτήσεις από άλλα προγράμματα ή/και ανταλλαγή δεδομένων.. 20

Βιβλιογραφία

Tulach, Jaroslav (2008). *Practical API Design: Confessions of a Java Framework Architect*. Apress.

