Περιεχόμενα

1 Τελική Εργασία Σεπτεμβρίου

Μερικές διευκρινίσεις, όσον αφορά τις απαιτήσεις της εργασίας και τον τρόπο που θα βαθμολογηθεί:

1. Το πιο σημαντικό είναι η προσέγγιση του προβλήματος με μεθόδους του Αντικειμενοστραφούς Προγραμματισμού, δηλαδή, χρησιμοποιήστε όσες και όποιες τεχνικές είδαμε κατά τη διάρκεια της χρονιάς για να προσεγγίσετε την πολυπλοκότητα του προγράμματος (Abstract classes, interfaces, inheritance)

2. Η εργασία που θα παραδώσετε θα πρέπει να μπορεί να γίνει compile και να εκτελεστεί (ακόμα και αν έχει λάθη / bugs / ατέλειες) ώστε να βαθμολογηθεί. Μπορείτε να χρησιμοποιήσετε όποια έκδοση της Java επιθυμείτε. Ωστόσο, από την έκδοση 9 και μετά η γλώσσα παρέχει λειτουργικότητα που θα σας διευκολύνει σημαντικά. Παραδώστε την εργασία, ακόμα και αν δεν έχετε υλοποιήσει όλες τις λειτουργίες που περιγράφονται.

3. Δώστε προσοχή στην εκφώνηση (και ρωτήστε όπου έχετε απορία), ώστε να μη χρειαστεί να λύσετε ένα αρκετά πιο δύσκολο πρόβλημα από αυτό της εργασίας (π.χ. ο χρόνος, στην εργασία, είναι διακριτός και "ορίζεται" από τις κινήσεις των μονάδων του παίκτη, αν προσπαθήσετε να προσεγγίσετε το πρόβλημα σε συνεχή χρόνο, είναι αρκετά πιο δύσκολο).

4. Χρησιμοποιήστε τόσο τα Java streams, όσο και τα utility methods στις συλλογές της Java, `List.of`, `Map.of`, ώστε να εξοικειωθείτε με αυτά και να ελαττώσετε τον κώδικα που θα χρειαστεί να γράψετε. Για παράδειγμα `for` που επιλέγει κάποια στοιχεία από μια συλλογή, μπορεί να αντικατασταθεί με ένα stream της μορφής `aList.stream().filter(...).collect(Collectors.toList())` και ένα κατάλληλο `lambda` expression στη μέθοδο filter.

5. Αναλύστε το πρόβλημα πριν ξεκινήσετε να γράφετε, και αν στην πορεία διαπιστώσετε ότι κάποια κλάση / σχεδιαστική απόφαση σας δυσκολεύει, μη διστάσετε να την αλλάξετε. Σκοπός είναι, εκτός των άλλων, να εξασκηθείτε.

6. Για την εργασία δεν απαιτείται η χρήση Threads, αλλά ούτε και απαγορεύεται. Επίσης μπορείτε να χρησιμοποιήσετε τα Design Patterns πού είδαμε στις δύο τελευταίες εργασίες (το Observer Pattern προσφέρεται ιδιαίτερα, καθώς επίσης και το Composite). Ωστόσο, μην το θεωρήσετε υποχρεωτικό.

7. Χρησιμοποιήστε ένα καλό περιβάλλον προγραμματισμού, ώστε να μπορείτε να αυτοματοποιήσετε κάποιες εργασίες (δημιουργία getters και setters, δημιουργία των `hash` και `equals`, `toString` κλπ).

8. Σε πάρα πολλά σημεία προτείνονται κάποιες προσεγγίσεις, αλλά η επιλογή αφήνεται σε εσάς. Στα σημεία αυτά (αλλά και στο παιχνίδι, γενικότερα) μπορείτε να ακολουθήσετε όποια προσέγγιση θέλετε.

9. Τέλος, υπάρχουν πάρα πολλοί τρόποι με τους οποίους μπορείτε να μοντελοποιήσετε το παιχνίδι. Ένα αρκετά καλό (αλλά όχι και το μοναδικό) κριτήριο για να δείτε κατά πόσο η μοντελοποίηση στην οποία καταλήξατε είναι καλή είναι να δείτε πόσο εύκολα μπορείτε να προσθέσετε νέους τύπους μονάδων και νέες δυνατότητες στις μονάδες αυτές.

Καλή επιτυχία!

2 Παιχνίδια Στρατηγικής (Turn-based)

Στα παιχνίδια στρατηγικής, είτε πραγματικού χρόνου, είτε turn-based, ο παίκτης χειρίζεται μια ή περισσότερες μονάδες (units), οι οποίες συνήθως έχουν συμπληρωματική λειτουργία, ως προς τις δυνατότητες και τα χαρακτηριστικά τους και μάχεται εναντία σε ένα ή περισσόερα αντίστοιχα σύνολα από μονάδες τα οποία ελέγχει είτε ο υπολογιστής, είτε κάποιος άλλος παίκτης.

Τόσο ο χώρος στον οποίο διεξάγονται οι μάχες, όσο και ο χρόνος στον οποίο λαμβάνουν χώρα οι κινήσεις (επιλογές ή actions) του παίκτη μπορεί να είναι είτε συνεχής είτε διακριτός. Συνήθως έχουμε δυο κατηγορίες:

- Συνεχής χώρος και συνεχής χρόνος, στην οποία ανήκουν τα Real-Time Strategy games (RTS),

- Διακριτός χώρος και διακριτός χρόνος, στην οποία ανήκουν τα Turn-Based Strategy games (TBS), η οποία είναι και η κατηγορία του παιχνιδιού που καλείστε να κατεασκευάσετε.

Στα παιχνίδια στρατηγικής οι παίκτες καλούνται να εκμεταλλευτούν και να οργανώσουν τις διαφορετικές δυνατότητες των μονάδων που ελέγχουν, καθώς και να εκμεταλλευτούν κατάλληλα τις τυχόν ιδιομορφίες του χώρου και (ενδεχομένως) των πόρων που είναι διαθέσιμοι στον παίκτη, ώστε να πετύχουν κάποιον σκοπό (ολική κυριαρχία στους αντιπάλους, κατάληψη συγκεκριμένων σημείων του χώρου, κ.α.).

Το παιχνίδι που θα υλοποιήσετε είναι μια αρκετά απλή μορφή ενός παιχνιδιού στρατηγικής, κατά την οποία ο παίκτης αντιμετωπίζει διαφορετικούς τύπους εχθρών σε μια σειρά από ”πίστες”. Τόσο οι μονάδες του παίκτη, όσο και οι αντίπαλοι έχουν κάποια βασικά κοινά χαρακτηριστικά, διαφέρουν ωστόσο ως προς τις δυνατότητές τους.

2.1 Γενική περιγραφή του παιχνιδιού

Σκοπός του παιχνιδιού είναι να βοηθήσετε τον γενναίο ιππότη Galahad, τους ιππότες-ιερείς του και τους τοξότες του να αποδράσουν από τη σκοτεινή περιοχή της Dinas Emrys, στην οποία κατοικούν πλάσματα της σκιάς και την οποία και φυλάνε δυο τρομεροί δράκοι. Ο Galahad πρέπει να επιβιώσει…

Ο κόσμος του παιχνιδιού αποτελείται από `N` διαφορετικές πίστες (`N >= 6`), σε κάθε μία από τις οποίες ο παίκτης αντιμετωπίζει διαφορετικούς αντιπάλους με διαδοχικά αυξανόμενη δυσκολία (και αντίστοιχες ανταμοιβές σε ”πόντους εμπειρίας” - experience points).

2.1.1 Χώρος διεξαγωγής των μαχών

Οι μάχες του παιχνιδιού διεξάγονται σε ένα πλέγμα που αποτελείται από εξαγωνικά πλακίδια. Οι διαστάσεις κάθε πίστας είναι (τουλάχιστον) 12 γραμμές επί 18 στήλες, αν και είσαστε ελεύθεροι να επιλέξετε κάποιο άλλο μέγεθος για κάποια από τις πίστες.

2.1.2 Χρόνος του παιχνιδιού

Ο χρόνος στο παιχνίδι είναι ”διακριτός” και οδηγείται από τις επιλογές του χρήστη. Πιο συγκεκριμένα, η ροή του παιχνιδιού αποτελείται από ”γύρους”. Κάθε γύρος ολοκληρώνεται όταν όλες οι μονάδες του παίκτη και όλες οι μονάδες που ελέγχονται από τον υπολογιστή εκτελέσουν κάποια ”δράση” (κίνηση ή επίθεση ή εκτέλεση κάποιου spell). Η διεξαγωγή των γύρων, είναι με τη σειρά της διακριτή. Ο παίκτης καλείται να επιλέξει κάποια δράση για κάθε μία από τις μονάδες που ελέγχει. Αφού ολοκληρωθεί η επιλογή δράσεων για τις μονάδες (οι δράσεις εκτελούνται τη στιγμή που επιλέγονται), ο υπολογιστής επιλέγει και εκτελεί δράσεις για τις μονάδες των αντιπάλων. Στο τέλος κάθε γύρου γίνονται οι απαραίτητοι έλεγχοι για το αν η πίστα ολοκληρώθηκε ή το παιχνίδι τελείωσε.

Μετά την ολοκλήρωση της κάθε πίστας, τα στατιστικά των μονάδων του παίκτη που υπάρχουν ακόμα στο παιχνίδι επανέρχονται στις μέγιστες τιμές τους.

2.1.3 Δράσεις των μονάδων του παίκτη

Η επιλογή των δράσεων των μονάδων του παίκτη μπορεί να γίνει με διάφορους τρόπους. Για ευκολία προτείνεται να γίνεται από το πληκτρολόγιο, ωστόσο, μπορείτε να επιλέξετε όποιον τρόπο θέλετε.

Οι δυνατές δράσεις των μονάδων είναι οι ακόλουθες:

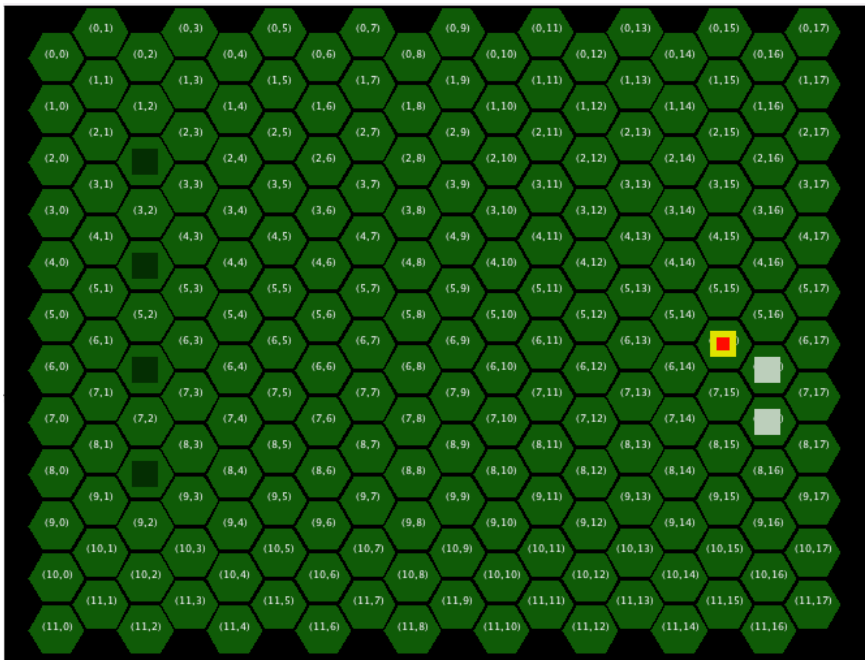
- Κίνηση σε κάποιο γειτονικό πλακίδιο. Επειδή στο εξαγωνικό πλέγμα του παιχνιδιού, κάθε πλακίδιο συνορεύει με (το πολύ) 6 άλλα και θα μπορούσατε,

για παράδειγμα, να χρησιμοποιήσετε τα πλήκτρα `Q`, `W`, `E`, `A`, `S`, `D` (ΒΔ, Β, ΒΑ, ΝΔ, Ν, ΝΑ).

- Επίθεση σε κάποιον αντίπαλο που βρίσκεται εντός ακτίνας δράσης της μονάδας. Μπορείτε να χρησιμοποιήσετε το πλήκτρο `X`
- Εκτέλεση κάποιου heal (αναπλήρωση hit points μιας μονάδας), μπορείτε να χρησιμοποιήσετε το πλήκτρο `H`
- Μετάβαση στην επόμενη μονάδα, χωρίς να εκτελεστεί κάποια δράση (pass). Μπορείτε να χρησιμοποιήσετε το πλήκτρο `SPACE`

3 Ο Χάρτης του παιχνιδιού

Ο χάρτης του παιχνιδιού αποτελείται από εξαγωνικά πλακίδια, τοποθετημένα σε γραμμές και στήλες. Υπάρχουν αρκετές προσεγγίσεις για το πως μπορείτε να υλοποιήσετε ένα εξαγωνικό πλέγμα, καθώς και το πως υπολογίζουμε γείτονες και αποστάσεις. Ενδεικτικά, δείτε τα σχετικό άρθρο στο site Red Blob Games, το οποίο περιλαμβάνει αρκετές διαφορετικές προσεγγίσεις.



Σχήμα 1: Το εξαγωνικό πλέγμα του παιχνιδιού

Στην εικόνα 1 βλέπουμε ένα τέτοιο πλέγμα στο οποίο είναι τοποθετημένοι 4 μονάδες που ελέγχει ο υπολογιστής και 3 μονάδες που ελέγχει ο παίκτης, ενώ ”σημειωμένη” με κόκκινο χρώμα είναι η μονάδα της οποίας είναι η σειρά να κινηθεί.

Στις υποδείξεις στο τέλος της εργασίας περιλαμβάνεται κάποιος ενδεικτικός κώδικας για τη δημιουργία του πλέγματος και τη σχεδίασή του σε έναν `Canvas` της Java. Ωστόσο, είστε ελεύθεροι να ακολουθήσετε όποια προσέγγιση θέλετε.

4 Μονάδες

Όλες οι μονάδες, είτε ελέγχονται από τον παίκτη, είτε ελέγχονται από τον υπολογιστή έχουν κάποια κοινά βασικά χαρακτηριστικά, πιο συγκεκριμένα:

- Όνομα: το όνομα του τύπου της μονάδας
- Maximum Hit Points: το μέγιστο πλήθος hit points της μονάδας, δηλαδή πόση συνολικά επίθεση μπορεί να δεχτεί πριν αφαιρεθεί από το παιχνίδι
- Current (actual) Hit Points: Τα hit points που έχει την εκάστοτε στιγμή
- Maximum Mana Points: Το mana είναι το resource που καταναλώνεται όταν οι μονάδες εκτελούν κάποιο spell – στην περίπτωσή μας το action `heal`, το οποίο και είναι διαθέσιμο στις μονάδες του παίκτη και έχει ένα σταθερό κόστος.
- Actual Mana Points: Τα mana points τα οποία έχει κάθε unit ανά πάσα στιγμή
- Attack Range: Η ακτίνα στην οποία μπορεί να επιτεθεί η μονάδα. Προτείνεται για ευκολία, τόσο στην περίπτωση των μονάδων του παίκτη, όσο και στην περίπτωση των μονάδων που ελέγχει ο υπολογιστής να επιλέγεται αυτόματα κάποια από τις μονάδες που βρίσκονται εντός της ακτίνας αυτής, είτε π.χ. με κριτήριο την απόσταση είτε με το πλήθος των current hit points, είτε τυχαία.
- Attack Power / Damage Type: Κάθε μονάδα έχει έναν τύπο επίθεσης και ένα attack power. Οι διαφορετικοί τύποι επίθεσης είναι οι ακόλουθοι
  - `PHYSICAL`
  - `ELEMENTAL_FIRE`
  - `ELEMENTAL_FROST`

  Για ευκολία θεωρήστε ότι το attack power καθορίζει και το ”healing power” κάθε μονάδας, δηλαδή το πόσα hit points αναπληρώνει κατά την εκτέλεση ενός heal.

- Armor Class: Η άμυνα που έχει η κάθε μονάδα στους διαφορετικούς τύπους επίθεσης. Το Armor Class του αμυνόμενου αφαιρείται από το Attack Power του επιτιθέμενου για τον συγκεκριμένο τύπο επίθεσης, χωρίς ωστόσο αυτή η διαφορά να μπορεί να γίνει μικρότερη του 1 (δηλαδή ο αμυνόμενος θα υποστεί τουλάχιστον 1 hit point damage).

Εκτός από τα παραπάνω, κάθε μονάδα πρέπει να είναι σε θέση να περιγράψει τους διαφορετικούς τύπου "action" τα οποία μπορει να εκτελέσει ο χρήστης, για παράδειγμα, στο συγκεκριμένο παιχνίδι, οι μονάδες που χειρίζεται ο υπολογιστής έχουν διαθέσιμο μόνο το action της επίθεσης, ενώ οι μονάδες του παίκτη έχουν, ανάλογα με τον τύπο τους διαθέσιμό τόσο action `ATTACK`, όσο και το action `HEAL`.

4.1 Μονάδες που χειρίζεται ο παίκτης

Οι μονάδες που χειρίζεται ο παίκτης έχουν κάποια επιπλέον χαρακτηριστικά και κάποιες επιπλέον δυνατότητες. Η πρώτη και βασικότερη είναι ότι έχουν τη δυνατότητα να δεχτούν heals από άλλα units (αν θέλετε μπορείτε να μοντελοποιήσετε αυτή τη δυνατότητα και στις μονάδες των αντιπάλων, και να προσθέσετε και άλλους τύπους units).

Η δεύτερη βασική διαφορά είναι ότι, σε αντίθεση με τα units των αντιπάλων συσσωρεύουν εμπειρία (experience points), η οποία και καταγράφει την πρόοδό τους και επηρεάζει τα βασικά χαρακτηριστικά τους. Κάθε μονάδα του υπολογιστή που νικιέται κατά τη διάρκεια ενός γύρου αποδίδει στις μονάδες του παίκτη ένα συγκεκριμένο "ποσό" από πόντους εμπειρίας, οι οποίοι και μεταβάλλουν τα βασικά χαρακτηριστικά των μονάδων του παίκτη.

Αν και υπάρχουν πολλες διαφορετικές προσεγγίσεις για το πως μπορεί να λειτουργεί το σύστημα των πόντων εμπειρίας, θα χρησιμοποιήσουμε την πιο απλή δυνατή μέθοδο, δηλαδή, τα χαρακτηριστικά των μονάδων παίκτη θα αυξάνονται ανάλογικά με τους πόντους εμπειρίας που έχει, πιο συγκεκριμένα `maxStat = baseStat + experiencePoints * statFactor`.

4.2 Actions

Υπάρχουν 2 διαφορετικά actions που μπορούν να εκτελέσουν τα units σε κάθε γύρο του παιχνιδιού, όταν είναι η σειρά τους (η κίνηση θεωρείται δεδομένη): `ATTACK` και `HEAL`.

Το `ATTACK` περιορίζεται από το `attack range` του εκάστοτε unit, ενώ το `HEAL` δεν έχει κάποιον περιορισμό στην απόσταση και επιδρά σε οποιοδήποτε unit.

Μια διαφορά από την εργασία του Ιουνίου είναι ότι το `ATTACK` μπορεί να έχει κάποιο `ATTACK-AREA` (η προεπιλεγμένη τιμή είναι `0`). Το attack area ορίζεται με βάση την απόσταση από κάποιο κεντρικό πλακίδιο. Αν το `ATTACK-AREA` είναι

`0`, τότε η επίθεση είναι συγκεντρωμένη σε ένα πλακίδιο. Για `ATTACK-AREA` ίσο με `1`, η επίθεση κάνει το **ίδιο** damage στο πλακίδιο που βρίσκεται ο στόχος και σε όλες τις **εχθρικές** μονάδες που βρίσκονται στα γειτονικά πλακίδια (απόσταση 1). Στην πραγματικότητα `ATTACK-AREA N` συμπεριλαμβάνει και όλα τα πλακίδια που βρίσκονται μέχρι και σε απόσταση `N` από το πλακίδιο στο οποίο βρίσκεται ο στόχος. Από προτεινόμενες μονάδες που παρουσιάζουμε στη συνέχεια, μόνο ο ”BlueDragon” έχει αυτή τη δυνατότητα.

Συμβουλή: Σκεφτείτε πως θα μπορούσατε να μοντελοποιήσετε τα actions, ώστε να είναι εύκολα επεκτάσιμα και τροποποιήσιμα.

Τα units που ελέγχει ο υπολογιστής είναι σε θέση να εκτελέσουν μόνο ένα από αυτά τα actions, το attack, ενώ τα units του παίκτη ενδέχεται να έχουν και τα 2 διαθέσιμα. Όσον αφορά το unit στο οποίο θα επιδράσουν αυτά τα actions, προτείνεται το εξής:

- heal: επιλέγεται αυτόματα το unit το οποίο έχει τη μεγαλύτερη διαφορά `max hit points - current-hit-points`
- attack: επιλέγεται το κοντινότερο Unit το οποίο βρίσκεται εντός του attack range ή ένα τυχαίο unit εντός αυτού.

4.3 Μονάδες και χαρακτηριστικά

Τα παρακάτω είναι ενδεικτικά – μπορείτε αν θέλετε να προσθέσετε / αλλάξετε τα ονόματα και τα χαρακτηριστικά των μονάδων.

4.3.1 Galahad

Ενδεικτικά στατιστικά του unit

- base hit points: 80
- base mana points: 0
- base attack power: 20
- attack range: 1
- actions: Attack / PHYSICAL, Heal
- armor class: Physical: 6, Frost: 3, Fire: 3
- experience modifiers: Hit points: 0.2, Mana points: 0.0, Armor class: 0.01, Attack power: 0.05

Ο Galahad, σε αντίθεση με τους Paladins που μπορούν να εκτέλούν `heal` όσο έχουν αρκετά mana points έχει ένα μοναδικό `heal` σε κάθε πίστα του παιχνιδιού, το οποίο επαναφέρει τα hit points μιας μονάδας στο μέγιστο (μπορείτε να το σκεφτείτε και σαν ένα Heal action το οποίο αυξάνει, π.χ. τα hit points κατά `Math.min(hitPoints + 10000, getMaxHitPoints())`)

4.3.2 Paladins

- base hit points: 40
- base mana points: 50
- base attack power: 15
- attack range: 1
- actions: Attack / PHYSICAL, Heal
- armor class: Physical: 6, Frost: 3, Fire: 3
- experience modifiers: Hit points: 0.1, Mana points: 0.4, Armor class: 0.01, Attack power: 0.05

4.3.3 Fire Archers

- base hit points: 10
- base mana points: 0
- base attack power: 20
- attack range: 6
- actions: Attack / $\text{ELEMENTAL}_{\text{FIRE}}$
- armor class: Physical: 1, Frost: 0, Fire: 10
- experience modifiers: Hit points: 0.1, Mana points: 0.0, Armor class: 0.01, Attack power: 0.05

4.3.4 Frost Archers

- base hit points: 10
- base mana points: 0
- base attack power: 20
- attack range: 6
- actions: Attack / $\text{ELEMENTAL}_{\text{FROST}}$
- armor class: Physical: 1, Frost: 0, Fire: 10
- experience modifiers: Hit points: 0.1, Mana points: 0.0, Armor class: 0.01, Attack power: 0.05

4.4 Αντίπαλες Μονάδες

Τα παρακάτω είναι ενδεικτικά – μπορείτε αν θέλετε να προσθέσετε / αλλάξετε τα ονόματα και τα χαρακτηριστικά των μονάδων.

4.4.1 Goblin

- base hit points: 30
- base mana points: 0
- base attack power: 20
- attack range: 1
- actions: Attack / PHYSICAL
- armor class: Physical: 5, Frost: 0, Fire: 0
- experience points: 25

4.4.2 Forest Troll

- base hit points: 50
- base mana points: 0
- base attack power: 20

- attack range: 1
- actions: Attack / PHYSICAL
- armor class: Physical: 5, Frost: 5, Fire: 0
- experience points: 50

4.4.3 Mountain Troll

- base hit points: 80
- base mana points: 0
- base attack power: 30
- attack range: 1
- actions: Attack / PHYSICAL
- armor class: Physical: 5, Frost: 8, Fire: 5
- experience points: 80

4.4.4 Wyvern

- base hit points: 120
- base mana points: 0
- base attack power: 35
- attack range: 1
- actions: Attack / $\text{ELEMENTAL}_{\text{FIRE}}$
- armor class: Physical: 8, Frost: 0, Fire: 1000
- experience points: 150

4.4.5 Red Dragon

- base hit points: 200
- base mana points: 0
- base attack power: 50
- attack range: 1
- actions: Attack / $\text{ELEMENTAL}_{\text{FIRE}}$
- armor class: Physical: 8, Frost: 0, Fire: 1000
- experience points: 1000

4.4.6 Blue Dragon

- base hit points: 200
- base mana points: 0
- base attack power: 50
- attack range: 1
- attack area: 1 κεντρικό + 6 γειτονικά πλακίδια
- actions: Attack / $\text{ELEMENTAL}_{\text{FROST}}$
- armor class: Physical: 8, Frost: 1000, Fire: 0
- experience points: 1000

4.5 Επιπλέον λειτουργικότητα που μπορείτε να υλοποιήσετε

- Εμπόδια: Ένα πλήθος από πλακίδια που είναι απροσπέλαστα από τις μονάδες
- Επιπλέον Units: Healers (μόνο Heal και όχι Attack)
- Πλακίδια που τροποποιούν τα στατιστικά της μονάδας που βρίσκεται πάνω τους

5 Υποδείξεις

Οι συγκεκριμένες υποδείξεις αποσκοπούν στο να σας βοηθήσουν, χωρίς, ωστόσο να είναι σε καμία περίπτωση υποχρεωτικό να τις ακολουθήσετε.

5.1 Γραφικά / Σχεδιασμός πίστας

5.1.1 Δημιουργία της πίστας

```
public class GameMap {
  /* ... */
  List<MapTile> tileList;
  Map<HexCoordinates, MapTile> coordinatesTileMap;
  int neighborDirectionsOddColumns[][] =  { {-1, -1}, { 0, -1},
    {1, -1}, {1, 0},
    {0, 1}, {-1, 0} };
  int neighborDirectionsEvenColumns[][] = { {-1,  0}, { 0, -1},
    {1, 0}, {1, 1},
    {0, 1}, {-1, 1} };


  private List<MapTile> initializeTiles(int rows, int cols) {
    var tmpArray = new MapTile[rows][cols];

    for(int r = 0; r < rows; r += 1) {
      for(int c = 0; c < cols; c += 1) {
tmpArray[r][c] = new MapTile(r, c);
      }
    }

    for(int r = 0; r < rows; r += 1) {
      for(int c = 0; c < cols; c += 1) {
int[][] neighborDirections;
if(c % 2 == 1) {
  neighborDirections = neighborDirectionsOddColumns;
} else {
  neighborDirections = neighborDirectionsEvenColumns;
}

for (int q = 0; q < neighborDirections.length; q++) {
  var nd = neighborDirections[q];
  var nRow = r + nd[1];
  var nCol = c + nd[0];
  if ((nRow >= 0) && (nRow < rows) && (nCol >= 0) && (nCol < cols)) {
    tmpArray[r][c].connect(tmpArray[nRow][nCol], q);
```

```
  }
}
      }
    }

    ArrayList<MapTile> retList = new ArrayList<>();
    for(int i = 0; i < rows; i += 1) {
      for(int j = 0; j < cols; j += 1) {
if(tmpArray[i][j] != null)
  retList.add(tmpArray[i][j]);
      }
    }
    return retList;
  }

  /* Ενδεικτική βοηθητική inner class για εύρεση των πλακιδίων εντός
   * μιας ακτίνας */
  public class DistancedTile {
    private final MapTile tile;
    private final int distance;
    /* Getters, Constructor, etc */
  }

  public List<DistancedTile> getTilesWithinRange(HexCoordinates hc,
    int distance) {
    /* Breadth first search */
  }
}
```

5.1.2 Σχεδιασμός της πίστας

```
public class DrawableMapTile implements Drawable {
  static final double deg30rad = 0.5;
  private final MapTile mapTile;
  public static final int TILE_CENTER_DIST = 40; // pixels
  public DrawableMapTile(MapTile mapTile) {
    this.mapTile = mapTile;
  }

  /** Επιστρέφει το κέντρο του HexTile ως προς την x συντεταγμένη */
```

```
 public double getScreenCenterX(int distanceOfTileCenters,
int offsX, int offsY) {
   int x = mapTile.getCol();
   int y = mapTile.getRow();
   double rLength = distanceOfTileCenters/2.0/Math.cos(deg30rad);
   double centerX = x * (1.5 * rLength) + offsX;
   return centerX;
 }

   /** Επιστρέφει το κέντρο του HexTile ως προς την y συντεταγμένη */
 public double getScreenCenterY(int distanceOfTileCenters,
int offsX, int offsY) {
   int x = mapTile.getCol();
   int y = mapTile.getRow();
   double yEx = 0.0;
   if(x % 2 == 1) {
     yEx = Math.sin(deg30rad)*distanceOfTileCenters;
   }
   double centerY = y * distanceOfTileCenters + offsY - yEx;
   return centerY;
 }

 /** Εξάγωνο με κέντρο το κέντρο του HexTile και "ακτίνα" distanceOfTileCe
 private Polygon getHexTilePoly(int distanceOfTileCenters,
int polyDist, int offsX, int offsY) {

   double centerX = getScreenCenterX(distanceOfTileCenters, offsX, offsY)
   double centerY = getScreenCenterY(distanceOfTileCenters, offsX, offsY)

   double paintLength = polyDist/2.0/Math.cos(deg30rad);

   // double[] angles = {0, 60, 120, 180, 240, 300}; // in radians
   double[] angles = {
     0.0, 1.0471975511965976,
     2.0943951023931953, 3.141592653589793,
     4.1887902047863905, 5.235987755982989
   };
   Polygon retP = new Polygon();
   for(int i = 0; i < 6; i++) {
     int px = (int) (centerX + Math.cos(angles[i]) * paintLength);
```

```
      int py = (int) (centerY + Math.sin(angles[i]) * paintLength);
      retP.addPoint(px, py);
    }
    return retP;
  }

  @Override
  public void draw(Graphics2D g, int offsX, int offsY) {
    g.setColor(Color.BLACK);
    g.fillPolygon(getHexTilePoly(TILE_CENTER_DIST, TILE_CENTER_DIST,
offsX, offsY));
    g.setColor(new Color(20, 80, 20));
    g.fillPolygon(getHexTilePoly(TILE_CENTER_DIST, TILE_CENTER_DIST-2,
offsX, offsY));

    // Fonts
    String coordString = "(" + mapTile.getRow() + "," + mapTile.getCol() +
    Font fnt = new Font(Font.SANS_SERIF, Font.PLAIN, 8);
    g.setFont(fnt);

    TextLayout textLayout = new TextLayout(coordString, fnt, g.getFontRende
    var stringBounds = textLayout.getBounds();

    g.setColor(Color.WHITE);
    g.drawString(coordString,
      (int) (getScreenCenterX(TILE_CENTER_DIST, offsX, offsY) -
stringBounds.getWidth()/2.0),
      (int) (getScreenCenterY(TILE_CENTER_DIST, offsX, offsY)));

  }
}
```

Σχεδιασμός των units του παίκτη (ενδεικτικός)

```
protected void paintPlayerUnits(Graphics2D g2d) {
     for(BaseHero bh: game.getPlayerUnits()) {
 HexCoordinates hx = bh.getLocation();
 DrawableMapTile dmt = new DrawableMapTile(game.getGameMap().getTileAt(hx)
 int tileDist = DrawableMapTile.TILE_CENTER_DIST;
 int xcenter = (int)dmt.getScreenCenterX(tileDist, OFFSX, OFFSY);
```

```
int ycenter = (int)dmt.getScreenCenterY(tileDist, OFFSX, OFFSY);
// Υποθέτουμε ότι κάθε Unit υλοποιεί ένα interface το οποίο
// μας επιστρέφει το χρώμα με το οποίο πρέπει να το
// απεικονίζουμε στην οθόνη -- προτιμήστε όποια υλοποίηση
// θέλετε.
g2d.setColor(bh.getColor());
g2d.fillRect(xcenter - tileDist/4, ycenter-tileDist/4,
  tileDist/2, tileDist/2);
if(bh == game.getCurrentHero()) {
    g2d.setColor(Color.RED);
    g2d.fillRect(xcenter - tileDist/8, ycenter-tileDist/8,
      tileDist/4, tileDist/4);
}
    }
}
```

5.2 Βασική οθόνη του παιχνιδιού

Η βασική οθόνη του παιχνιδιού μπορεί να αποτελείται από έναν `Canvas` στον οποίο και σχεδιάζεται η πίστα, ένα `JTextArea` στο οποίο καταγράφονται πληροφορίες για το παιχνίδι, καθως και μια περιοχή (`JLabel`) στην οποία καταγράφονται πληροφορίες για τις μονάδες του χρήστη (βλ. εικόνα).

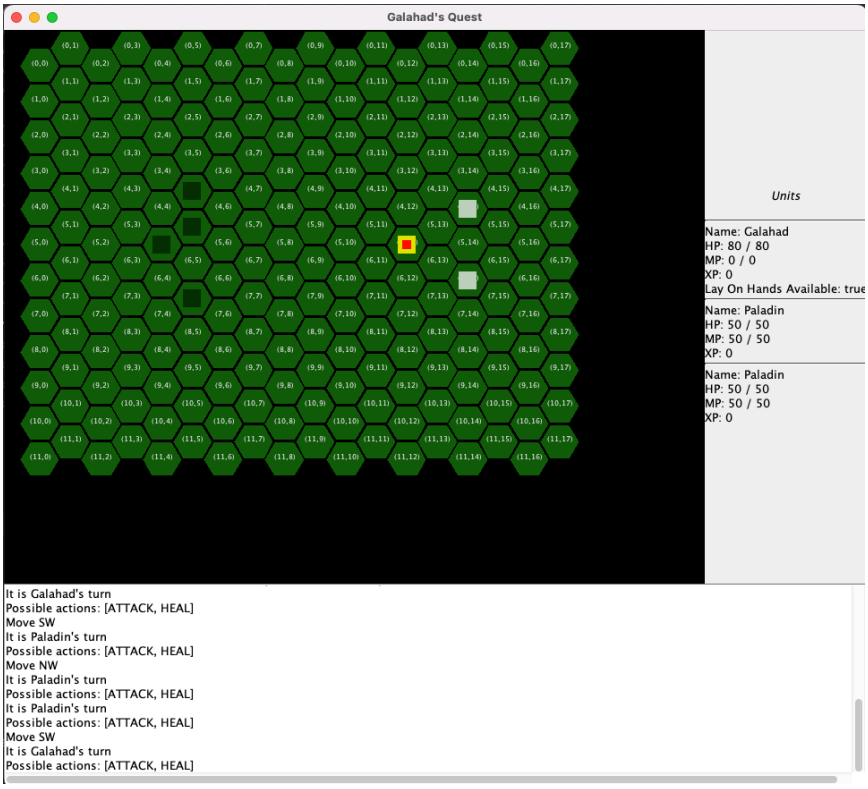
Όσον αφορά το `JTextArea` και το `JLabel` μπορείτε να χρησιμοποιήσετε κάποιο design pattern (observer) ή κάποια αντίστοιχη προσέγγιση. Ενδεικτικά, μια γενική μορφή της βασικής οθόνης του παιχνιδιού θα μπορούσε να είναι η ακόλουθη:

```
public GameFrame(Game game) {
     super("Galahad's Quest");
     JPanel gamePanel = new JPanel(); // Το βασικό Panel

     var layout = new BorderLayout();
     gamePanel.setLayout(layout);

     // Το subclass της Canvas που είναι υπεύθυνο για τη σχεδίαση του παιχ
     var mapFrame = new MapFrame(game);
     gamePanel.add(mapFrame, BorderLayout.CENTER);

     // ορίζεται αλλού -- το JLabel με τα στατιστικά των μονάδων
     // του παίκτη
     heroStatusLabel = new JLabel();
```



Σχήμα 2: Η βασική ”οθόνη” του παιχνιδιού

```
    heroStatusLabel.setText("<html><center><i>Units</i></center><br/></htm
    gamePanel.add(heroStatusLabel, BorderLayout.EAST);


    // Το βασικό παράθυρο πληροφοριών για το τι γίνεται στο παιχνίδι
    transcriptText = new JTextArea(14, 80);
    transcriptText.setEditable(false);
    JScrollPane scrollPane = new JScrollPane(transcriptText);
    game.addTranscriptTarget(this);
    gamePanel.add(scrollPane, BorderLayout.SOUTH);

    add(gamePanel);
    setVisible(true);
    pack();
    setDefaultCloseOperation(EXIT_ON_CLOSE);

    // Καλεί κάποια μέθοδο (στην προκειμένη περίπτωση "playerAction")
    // με παράμετρο το input (action) του χρήστη
    mapFrame.addKeyListener(new KeyAdapter() {
@Override
public void keyTyped(KeyEvent e) {
    game.playerAction(e.getKeyChar());
    mapFrame.repaint();
}
    });

    // Χρειάζεται, ώστε να επιστρέφεται το Focus στο βασικό panel
    // -- Ακόμα καλύτερα, θα πρέπει να επιστρέφει το Focus στην
    // κλάση που του MapFrame (Canvas)
    this.addWindowFocusListener(new WindowAdapter() {
public void windowGainedFocus(WindowEvent e) {
    gamePanel.requestFocusInWindow();
}
    });

    //
    game.addObserver(this);

    game.writeTranscript("Welcome to the trials of Galahad and his band o
    game.writeTranscript("Fierce battles await you!");
```

```
     // Set initial focus
     gamePanel.requestFocusInWindow();
 }
```

Υπενθυμίζουμε ότι το `JLabel` υποστηρίζει HTML, επομένως μπορείτε εύκολα να μορφοποιήσετε τα στατιστικά / χαρακτηριστικά των units. Για παράδειγμα, αν η κλάση του βασικού interface και η βάση του παιχνιδιού χρησιμοποιούν το observer pattern, ώστε να καλείται μια μέθοδος update του interface, όταν αλλάζει κάτι στο state των units του παίκτη, τότε, η `update` αυτή θα μπορούσε να έχει τη μορφή:

```
@Override
public void update(List<BaseHero> heroList) {
    var sbResult = new StringBuilder();
    sbResult.append("<html>")
    .append("<center><i>Units</i></center><br/>");
    for(BaseHero bh: heroList) {
sbResult.append(bh.getHTMLDescription());
    }
    sbResult.append("</html>");
    // heroStatusLabel είναι το JLabel
    heroStatusLabel.setText(sbResult.toString());
}
```

και αντίστοιχα, η `getHTMLDescription` (θα μπορούσε να είναι κάποιο `interface`, π.χ. `HTMLDescribeable`)

```
@Override
public String getHTMLDescription() {
    return new StringBuilder()
.append("<hr/>")
.append("Name: ").append(getName()).append("<br/>")
.append("HP: ").append(getCurrentHitPoints()).append(" / ").append(getMaxHi
.append("MP: ").append(getCurrentManaPoints()).append(" / ").append(getMaxM
.append("XP: ").append(getExperiencePoints())
.toString();
}
```

5.2.1 Αλληλεπίδραση του παίκτη με το περιβάλλον

Αν και συνήθως στα παιχνίδια στρατηγικής ο παίκτης αλληλεπιδρά με τον κόσμο και τις μονάδες κάνοντας κυρίως χρήση του ποντικιού, στο συγκεκριμένο παιχνίδι μπορείτε να χρησιμοποιήσετε μόνο το πληκτρολόγιο. Αυτός είναι και ένας

λόγος που στα στοιχεία του παιχνιδιού η μέγιστη απόσταση που μπορεί να κινηθεί κάθε μονάδα είναι 1 πλακίδιο και όχι παραπάνω.

5.3 Γενικές ιδέες για τον σχεδιασμό

Τα παρακάτω είναι κάποιες γενικές ιδέες. Μπορείτε να τις χρησιμοποιήσετε όπως θέλετε και δεν είναι σε καμία περίπτωση υποχρεωτικό να βασιστείτε σε αυτές

- Οι δράσεις / επιλογές που μπορεί να έχει ο παίκτης για κάθε μονάδα θα μπορούσαν να είναι μια κλάση ή ένα enum
- Το ίδιο και οι τύποι επίθεσης, καθώς και οι τύποι Armor Class
- Τα Units τα οποία μπορούν να δεχτούν damage, θα μπορούσαν να υλοποιούν κάποιο interface (Attackable?). Αντίστοιχα τα units που μπορούν να δεχτούν heals θα μπορούσαν να υλοποιούν κάποιο interface (Healable?). Αντίστοιχα, τα units τα οποία είναι σε θέση να εκτελέσουν heals, θα μπορούσαν να υλοποιούν το ”Healer” / ”Caster” interface.
- Τα Units έχουν κάποια κοινά βασικά χαρακτηριστικά, τα οποία χωρίζονται σε 2 κατηγορίες (Hero units & Enemy units) και τα οποία στη συνέχεια λαμβάνουν επιπλέον specializations.