ΔΙΕΘΝΕΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΤΗΣ ΕΛΛΑΔΟΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

«*Δημιουργία Παιχνιδιού Χαλαρωτικό Παζλ.*»

«Εικόνα»

Του φοιτητή
Τσενεκίδης Κων/νος.
Αρ. Μητρώου: 134086

Επιβλέπων
Γουλιάνας Κων/νος
Βαθμίδα ……

Ημερομηνία ……

Τίτλος Δ.Ε. ….
Κωδικός Δ.Ε. …
Ονοματεπώνυμο φοιτητή Τσενεκίδης Κων/νος
Ονοματεπώνυμο εισηγητή Γουλιάνας Κων/νος
Ημερομηνία ανάληψης Δ.Ε. …
Ημερομηνία περάτωσης Δ.Ε. …

Βεβαιώνω ότι είμαι ο συγγραφέας αυτής της εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, έχω καταγράψει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών, εικόνων και κειμένου, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επιπλέον, βεβαιώνω ότι αυτή η εργασία προετοιμάστηκε από εμένα προσωπικά, ειδικά ως διπλωματική εργασία, στο Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του ΔΙ.ΠΑ.Ε.

Η παρούσα εργασία αποτελεί πνευματική ιδιοκτησία των φοιτητού Κων/νου Τσενεκίδη που την εκπόνησε. Στο πλαίσιο της πολιτικής ανοικτής πρόσβασης, ο συγγραφέας/δημιουργός εκχωρεί στο Διεθνές Πανεπιστήμιο της Ελλάδος άδεια χρήσης του δικαιώματος αναπαραγωγής, δανεισμού, παρουσίασης στο κοινό και ψηφιακής διάχυσης της εργασίας διεθνώς, σε ηλεκτρονική μορφή και σε οποιοδήποτε μέσο, για διδακτικούς και ερευνητικούς σκοπούς, άνευ ανταλλάγματος. Η ανοικτή πρόσβαση στο πλήρες κείμενο της εργασίας, δεν σημαίνει καθ' οιονδήποτε τρόπο παραχώρηση δικαιωμάτων διανοητικής ιδιοκτησίας του συγγραφέα/δημιουργού, ούτε επιτρέπει την αναπαραγωγή, αναδημοσίευση, αντιγραφή, πώληση, εμπορική χρήση, διανομή, έκδοση, μεταφόρτωση (downloading), ανάρτηση (uploading), μετάφραση, τροποποίηση με οποιονδήποτε τρόπο, τμηματικά ή περιληπτικά της εργασίας, χωρίς τη ρητή προηγούμενη έγγραφη συναίνεση του συγγραφέα/δημιουργού.

Η έγκριση της διπλωματικής εργασίας από το Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του Διεθνούς Πανεπιστημίου της Ελλάδος, δεν υποδηλώνει απαραιτήτως και αποδοχή των απόψεων του συγγραφέα, εκ μέρους του Τμήματος.

«Αφιέρωση»

Πρόλογος

Η εργασία αυτή συντάχθηκε στο πλαίσιο της ολοκλήρωσης των προπτυχιακών σπουδών μου, με στόχο τη σύνδεση των θεωρητικών γνώσεων που αποκτήθηκαν κατά τη διάρκεια της φοίτησής μου με την πρακτική εφαρμογή τους μέσω της υλοποίησης μιας λειτουργικής και σύγχρονης διαδραστικής εφαρμογής.

Η επιλογή του θέματος βασίστηκε στο προσωπικό ενδιαφέρον για την ανάπτυξη παιχνιδιών, τις εφαρμογές εκπαίδευσης μέσω παιχνιδιού (edutainment), αλλά και τη χρήση μοντέρνων τεχνολογιών frontend. Η υλοποίηση του χαλαρωτικού παζλ αποτέλεσε πρόκληση και ευκαιρία ταυτόχρονα, καθώς περιλάμβανε πτυχές προγραμματισμού, αλγοριθμικής σκέψης, οπτικής σχεδίασης και τεκμηρίωσης.Θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή μου για την καθοδήγησή του, καθώς και όλους όσοι με υποστήριξαν κατά την εκπόνηση της παρούσας εργασίας.

Περίληψη

Η παρούσα πτυχιακή εργασία παρουσιάζει την ανάπτυξη μιας διαδραστικής εφαρμογής παιχνιδιού τύπου σταυρόλεξου, η οποία απευθύνεται σε ελληνόφωνους χρήστες. Το παιχνίδι λειτουργεί στον browser και έχει υλοποιηθεί με τη χρήση JavaScript και του γραφικού περιβάλλοντος Pixi.js. Κάθε επίπεδο δημιουργείται δυναμικά, στηριζόμενο σε αλγοριθμική λογική που επιλέγει τυχαία γράμματα και δημιουργεί λέξεις από λεξικό. Ο χρήστης καλείται να ανακαλύψει τις λέξεις του σταυρόλεξου ή να εντοπίσει επιπλέον λέξεις για πόντους.

Η εφαρμογή υποστηρίζει διαφορετικά επίπεδα δυσκολίας και bonus γύρους, προωθώντας τη συνεχή ενασχόληση του παίκτη. Εστιάζει τόσο στην ψυχαγωγία όσο και στην ενίσχυση των γλωσσικών δεξιοτήτων. Η εργασία καλύπτει την τεχνολογική υλοποίηση, την αρχιτεκτονική του project, τις προκλήσεις της ανάπτυξης, καθώς και τις προτάσεις για μελλοντική επέκταση.

«*Δημιουργία Παιχνιδιού Χαλαρωτικό Παζλ*»

(Relaxing Puzzle)

« Τσενεκίδης Κωνσταντίνος»

(Tsenekidis Konstantinos)

Abstract

This thesis was written in the context of completing my undergraduate studies, with the aim of connecting the theoretical knowledge acquired during my studies with their practical application through the implementation of a functional and modern interactive application.

The choice of the topic was based on my personal interest in game development, edutainment applications, and the use of modern frontend technologies. The implementation of the relaxing puzzle was both a challenge and an opportunity, as it included aspects of programming, algorithmic thinking, visual design, and documentation. I would like to thank my supervisor for his guidance, as well as everyone who supported me during the preparation of this thesis.

Ευχαριστίες

Θα ήθελα να εκφράσω τις ειλικρινείς μου ευχαριστίες στον επιβλέποντα καθηγητή μου, Γουλιάνα Κωνσταντίνο, για την πολύτιμη καθοδήγηση, την υποστήριξη και τις εποικοδομητικές παρατηρήσεις καθ' όλη τη διάρκεια της εκπόνησης της παρούσας πτυχιακής εργασίας. Η εμπειρία και η διάθεσή του για συνεργασία συνέβαλαν ουσιαστικά στην ολοκλήρωση του έργου αυτού.

Επίσης, ευχαριστώ θερμά την οικογένειά μου για την αδιάκοπη ηθική στήριξη, την υπομονή και την ενθάρρυνση σε κάθε στάδιο των σπουδών μου.

Τέλος, θα ήθελα να ευχαριστήσω τους φίλους και συμφοιτητές μου για τις δημιουργικές ανταλλαγές απόψεων και τη βοήθεια που μου προσέφεραν όταν τη χρειάστηκα.

Περιεχόμενα

Κατάλογος Σχημάτων

Συντομογραφίες

Δ.Ε.	Διπλωματική Εργασία
ΔΙΠΑΕ	Διεθνές Πανεπιστήμιο Ελλάδος
Π.Ε.	Πτυχιακή Εργασία

Κεφάλαιο 1ο: Εισαγωγή

1.1 Εισαγωγή

Η ταχύτατη πρόοδος της τεχνολογίας και η ευρεία διάδοση του διαδικτύου έχουν δημιουργήσει νέα δεδομένα στην εκπαίδευση, την ψυχαγωγία και τη δημιουργία ψηφιακών εφαρμογών. Στο πλαίσιο αυτό, τα ψηφιακά παιχνίδια κατέχουν πλέον έναν ουσιαστικό ρόλο, όχι μόνο ως μέσα διασκέδασης, αλλά και ως εργαλεία μάθησης και ανάπτυξης δεξιοτήτων. Τα παιχνίδια λέξεων, ειδικότερα, αποτελούν διαχρονικά ένα σημαντικό μέσο εξάσκησης του λεξιλογίου, της παρατηρητικότητας και της λογικής σκέψης.

Στο παρόν πλαίσιο, η παρούσα πτυχιακή εργασία εστιάζει στην ανάπτυξη ενός διαδραστικού παιχνιδιού τύπου σταυρόλεξου, ειδικά σχεδιασμένου για την ελληνική γλώσσα. Το παιχνίδι λειτουργεί αποκλειστικά μέσω φυλλομετρητή (browser), χωρίς την ανάγκη εγκατάστασης, και στοχεύει τόσο στην ψυχαγωγία όσο και στη γλωσσική ενίσχυση του χρήστη. Με έμφαση στη δυναμική παραγωγή περιεχομένου, κάθε επίπεδο του παιχνιδιού είναι μοναδικό, δίνοντας στο χρήστη την αίσθηση της ανανέωσης και της προκλητικότητας.

Επιπλέον, η εφαρμογή εμπλουτίστηκε με σημαντικά χαρακτηριστικά όπως η προσθήκη αρχικού μενού για την επιλογή τρόπου παιχνιδιού, η υποστήριξη bonus γύρων με ειδική λογική ανταμοιβής, καθώς και η χρήση διαμαντιών για την απόκτηση βοήθειας. Οι λέξεις κατηγοριοποιούνται πλέον σε κύριες, έξτρα και λεξικού, ενώ ο μηχανισμός επιπέδων υποστηρίζει τόσο δυναμική παραγωγή όσο και στατικά προκαθορισμένα επίπεδα. Τέλος, αναπτύχθηκε script σε γλώσσα Python για την αυτόματη εξαγωγή καθαρού λεξικού λέξεων βάσει συχνότητας χρήσης στην ελληνική γλώσσα.

1.2 Σημασία των παιχνιδιών-λέξεων στην εκπαίδευση

Η χρήση των λεξιλογικών παιχνιδιών ως παιδαγωγικό εργαλείο είναι γνωστή εδώ και δεκαετίες. Πέρα από το στοιχείο της διασκέδασης, τα παιχνίδια αυτά ενισχύουν τη μνήμη, τη σύνδεση ήχων και γραμμάτων, τη σύνθεση λέξεων, και προσφέρουν δημιουργικούς τρόπους εκμάθησης. Στην ελληνική γλώσσα, η οποία διαθέτει πλούσιο λεξιλόγιο και πολύπλοκη μορφολογία, τέτοιου είδους παιχνίδια είναι ιδιαίτερα χρήσιμα για μαθητές και ενήλικες. Η εργασία αυτή αξιοποιεί τις δυνατότητες της **JavaScript** και των εργαλείων γραφικής αναπαράστασης (**Pixi.js**) ώστε να υλοποιήσει ένα σύγχρονο, καλοσχεδιασμένο παιχνίδι λέξεων που ανανεώνεται αυτόματα και διατηρεί το ενδιαφέρον του χρήστη.Επίσης χρησιμοποιώντας τις δυνατότητες της Python χειριζομαστε τις ελληνικές λέξεις για το λεξικό μας.

1.3 Κίνητρα επιλογής θέματος

Η επιλογή υλοποίησης ενός παιχνιδιού σταυρόλεξου βασίστηκε σε μια σειρά από τεχνολογικά και παιδαγωγικά κριτήρια:

- Η ανάγκη για δημιουργία μιας εφαρμογής με απλό και καθαρό gameplay.

- Η πρόκληση ανάπτυξης δυναμικού περιεχομένου μέσω αλγορίθμων, χωρίς στατικά δεδομένα.
- Η επιθυμία κατασκευής ενός εργαλείου που θα μπορεί να χρησιμοποιηθεί και εκπαιδευτικά.
- Η προτίμηση για τεχνολογίες frontend που δεν απαιτούν server-side εξαρτήσεις.

Το σταυρόλεξο, σε συνδυασμό με μια παλέτα από γράμματα, δίνει την ευκαιρία στον χρήστη να πειραματιστεί, να συνθέσει, και να ανακαλύψει λέξεις που υπάρχουν στο λεξικό. Η προσέγγιση αυτή ενισχύει και τον γλωσσικό πλούτο του παίκτη.

1.4 Τεχνικές προκλήσεις και σχεδιαστικές επιλογές

Κατά την ανάπτυξη της εφαρμογής, προέκυψαν μια σειρά από τεχνικές προκλήσεις:

- Πώς θα εξασφαλιστεί ότι κάθε σετ γραμμάτων θα μπορεί να σχηματίσει έγκυρες λέξεις;
- Πώς θα κατασκευαστεί ένα σταυρόλεξο σε πλέγμα χωρίς να επικαλύπτονται λάθος γράμματα;
- Πώς θα γίνεται σωστά η διάκριση λέξεων που χρησιμοποιούνται στο grid και λέξεων που λειτουργούν ως έξτρα (bonus);
- Πώς θα παραμείνει η εμπειρία του χρήστη απλή, χωρίς να απαιτεί περίπλοκα μενού;

Αυτά τα ερωτήματα απαντήθηκαν με τη δημιουργία ενός modular κώδικα, κατανεμημένου σε αρχεία ανά ρόλο (**game logic, level generation, UI scenes** κ.λπ.) και με την αξιοποίηση της **Pixi.js** για τη βελτιστοποίηση του rendering.

1.5 Σκοπός και στόχοι της εργασίας

Ο γενικός σκοπός της εργασίας είναι η ανάπτυξη ενός browser-based παιχνιδιού σταυρόλεξου στα ελληνικά, με δυναμική παραγωγή επιπέδων και δυνατότητες επέκτασης. Οι επιμέρους στόχοι περιλαμβάνουν:

- Την υλοποίηση λειτουργικού frontend με JavaScript και Pixi.js
- Τη δημιουργία αλγορίθμων που επιλέγουν λέξεις και τις τοποθετούν σε grid
- Τη σύνδεση με ελληνικό λεξικό μέσω Python script
- Τον σχεδιασμό μιας απλής, φιλικής και αποκριτικής διεπαφής χρήστη
- Την ενσωμάτωση στοιχείων επιβράβευσης (πόντοι, bonus, διαμάντια)

1.6 Δομή της Εργασίας

Η εργασία διαρθρώνεται σε εννέα βασικά κεφάλαια που καλύπτουν πλήρως τη διαδικασία ανάπτυξης της εφαρμογής. Αρχικά, στο πρώτο κεφάλαιο παρουσιάζεται η εισαγωγή στο θέμα και αναλύονται τα κίνητρα, οι στόχοι, οι προκλήσεις και το πλαίσιο της εργασίας. Το δεύτερο κεφάλαιο εστιάζει στην αναλυτική περιγραφή του project, τον τρόπο λειτουργίας του παιχνιδιού, τη δομή του gameplay και τα χαρακτηριστικά του χρήστη.

Στο τρίτο κεφάλαιο γίνεται λεπτομερής ανάλυση των τεχνολογιών που χρησιμοποιήθηκαν, όπως η **JavaScript**, το **Pixi.js** και το λεξικό. Το τέταρτο κεφάλαιο ασχολείται με την αρχιτεκτονική της εφαρμογής, περιγράφοντας τα κύρια αρχεία, τις βασικές μονάδες κώδικα και τη μεταξύ τους συνεργασία.Ακολουθεί το πέμπτο κεφάλαιο, όπου αναλύεται η διεπαφή χρήστη, η λειτουργία κάθε σκηνής, και η αλληλεπίδραση του παίκτη με το σύστημα. Το έκτο κεφάλαιο παρέχει οδηγίες για την

εγκατάσταση και την εκτέλεση της εφαρμογής σε τοπικό περιβάλλον, ενώ το έβδομο εξετάζει τις δυνατότητες παραμετροποίησης και μελλοντικής επεκτασιμότητας του project.

Στο όγδοο κεφάλαιο παρουσιάζονται τα συμπεράσματα που προέκυψαν από την υλοποίηση της εφαρμογής, οι εμπειρίες που αποκόμισε ο δημιουργός και οι προτάσεις για αξιοποίηση σε άλλες χρήσεις. Τέλος, το ένατο κεφάλαιο περιλαμβάνει το παράρτημα, με αποσπάσματα κώδικα, τεχνικά παραδείγματα και εικόνες από τη λειτουργία της εφαρμογής. Η εφαρμογή που αναπτύχθηκε στο πλαίσιο της παρούσας πτυχιακής εργασίας είναι ένα διαδραστικό χαλαρωτικό παιχνίδι τύπου σταυρόλεξου, σχεδιασμένο για χρήση μέσω browser, χωρίς την ανάγκη εγκατάστασης. Ο βασικός της στόχος είναι να προσφέρει μια εμπειρία μάθησης και ψυχαγωγίας, επιτρέποντας στους χρήστες να εξασκούν το λεξιλόγιο και την ορθογραφία τους, χρησιμοποιώντας ένα φιλικό και διαδραστικό περιβάλλον.

Η λειτουργία του παιχνιδιού βασίζεται σε έναν αλγόριθμο που:

- Επιλέγει τυχαία σύνολα γραμμάτων με βάση τη συχνότητα εμφάνισής τους στην ελληνική γλώσσα.
- Φιλτράρει το περιεχόμενο του λεξικού για να εντοπίσει τις λέξεις που μπορούν να σχηματιστούν με τα επιλεγμένα γράμματα.
- Δημιουργεί ένα πλέγμα τύπου σταυρολέξου, προσπαθώντας να τοποθετήσει όσες περισσότερες από αυτές τις λέξεις είναι εφικτό.
- Εμφανίζει στον παίκτη ένα περιβάλλον όπου καλείται να ανακαλύψει τις λέξεις του πλέγματος με τη σωστή σειρά χαρακτήρων.

Η διαδικασία αυτή δεν παράγει προκαθορισμένα επίπεδα, αλλά επιτρέπει στο παιχνίδι να δημιουργεί δυναμικά και μοναδικά επίπεδα κάθε φορά που παίζεται. Η χρήση του λεξικού μέσω αρχείου JSON επιτρέπει την εύκολη αντικατάσταση ή επέκτασή του, ενώ οι συναρτήσεις JavaScript διασφαλίζουν ότι μόνο οι έγκυρες λέξεις, ως προς τα διαθέσιμα γράμματα, περιλαμβάνονται στο σταυρόλεξο.

Κάθε επίπεδο περιλαμβάνει:

- Την παλέτα των διαθέσιμων γραμμάτων (letter palette)
- Το σταυρόλεξο (grid) με κενά γράμματα για να συμπληρωθούν
- Σημεία βοήθειας (κουμπιά hint)
- Δείκτες πόντων και επιπέδου

Επιπλέον χαρακτηριστικό του παιχνιδιού είναι τα bonus levels. Κάθε τρίτο επίπεδο είναι ένας "γύρος πρόκλησης", στον οποίο εμφανίζονται γράμματα και ο παίκτης καλείται να ανακαλύψει μια συγκεκριμένη λέξη μεγάλου μήκους, γνωστή ως "target word". Η επιτυχής εύρεση της λέξης αυτής προσφέρει διαμάντια, τα οποία μπορούν να χρησιμοποιηθούν για να αποκαλυφθούν γράμματα σε μελλοντικά επίπεδα ή για άλλα πλεονεκτήματα.

Το παιχνίδι χρησιμοποιεί:

- **Οθόνη αρχικής εισόδου** (start screen) με κουμπί "Παίξε τώρα"
- **Εναλλαγή μεταξύ σκηνών** (scenes): επιπέδου, bonus, ολοκλήρωσης, κ.λπ.
- **Φιλικό προς τον χρήστη interface** με animations, χρώματα και ξεκάθαρα κουμπιά επιλογών

Χάρη στη χρήση του Pixi.js, η εφαρμογή εκμεταλλεύεται πλήρως τον HTML καμβά (canvas), με ταχύτατη απόδοση και δυνατότητα εκτέλεσης ακόμα και σε φορητές συσκευές.

Η λογική πίσω από τη δημιουργία επιπέδων βρίσκεται κυρίως στο αρχείο LevelGenerator.js. Σε κάθε νέα προσπάθεια δημιουργίας επιπέδου:

- Γίνεται προσπάθεια να παραχθεί ένα σετ γραμμάτων που να οδηγεί σε τουλάχιστον 3 έγκυρες λέξεις
- Χρησιμοποιείται η συνάρτηση generateWords() για να εντοπιστούν οι δυνατές λέξεις
- Χρησιμοποιείται η συνάρτηση generateCrossword() για να τοποθετηθούν αυτές οι λέξεις σε πλέγμα
- Όσες λέξεις δεν χωρούν, καταγράφονται ως extraWords και προσφέρουν επιπλέον πόντους εάν εντοπιστούν από τον παίκτη

Η συνολική εμπειρία που παρέχει η εφαρμογή είναι ιδιαίτερα ελκυστική για τον χρήστη, καθώς συνδυάζει την τυπική μηχανική παιχνιδιού σταυρόλεξου με τη δυναμική αλληλεπίδραση και τον οπτικά ευχάριστο σχεδιασμό. Ενθαρρύνει τον παίκτη να βελτιώσει τις γλωσσικές του δεξιότητες μέσα από την πρόκληση και τη συνεχή ανανέωση των επιπέδων.

Κεφάλαιο 2ο: Παρουσίαση του “Crossword Puzzle Free”

Για την ανάπτυξη της εφαρμογής σταυρολέξου άντλησα έμπνευση από το δημοφιλές mobile παιχνίδι “Crossword Puzzle Free” της SoftTowel Games, διαθέσιμο σε Android και iOS. Στην παρούσα ενότητα θα παρουσιάσω το περιβάλλον, τις βασικές λειτουργίες και τα βοηθήματα του πρωτότυπου τίτλου, και θα δείξουμε πώς ενσωματώσαμε αντίστοιχες ιδέες στη δική μας υλοποίηση.

2.1 Κεντρικό Μενού

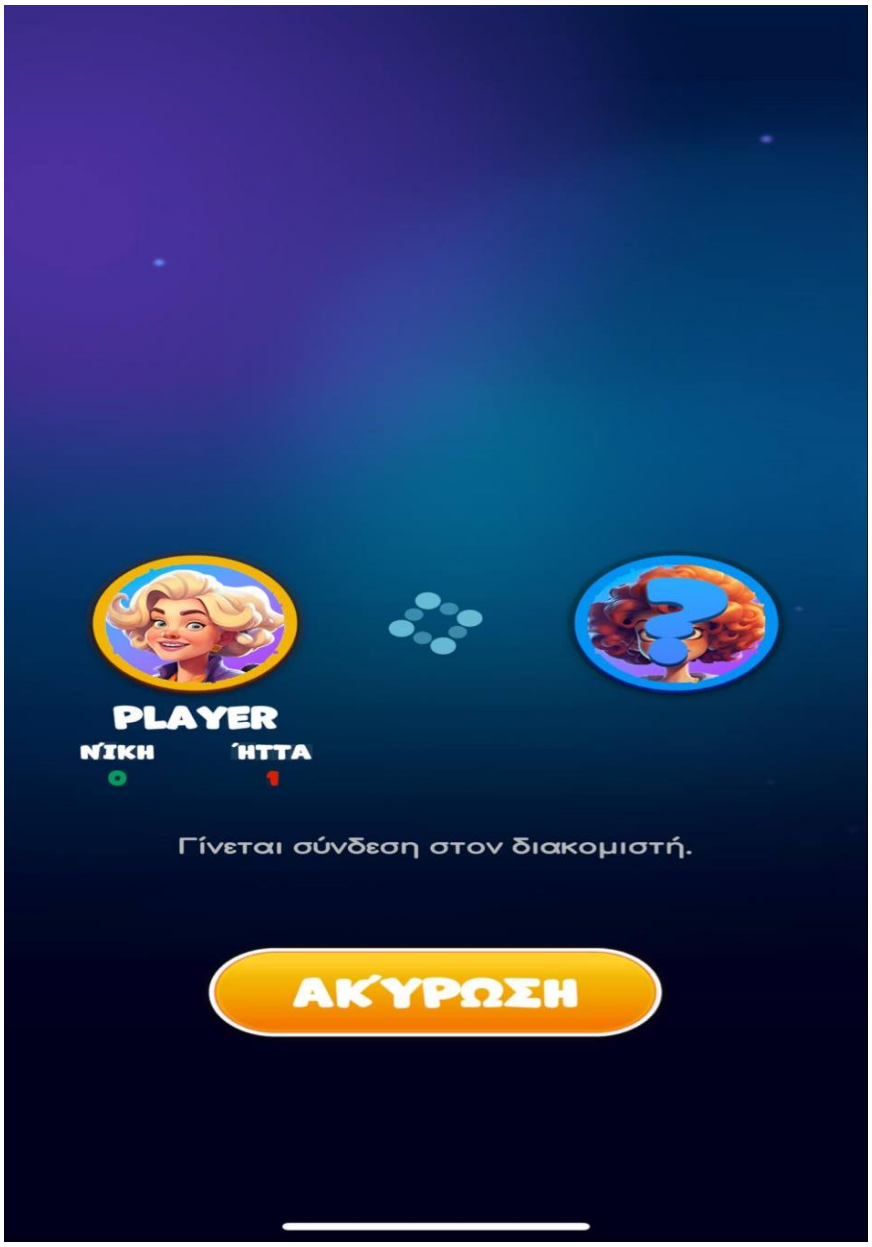
Στην κεντρική οθόνη του “Crossword Puzzle Free” ο παίκτης επιλέγει να ξεκινήσει από το επίπεδο που έχει σταματήσει ή να παίξει online με έναν τυχαίο παίκτη για το ποιος θα βρει τις περισσότερες λέξεις και πιο γρήγορα.

Εικόνα 1: Αρχικό μενού Crossword Puzzle Free

Στην παραπάνω εικόνα βλέπουμε το κουμπί με την επιλογή του επιπέδου που σταμάτησε ο χρήστης και το κουμπί multiplayer για παιχνίδι με τυχαίο χρήστη. Πιο πάνω υπάρχει ένας ήλιο ο οποίος αν το πατήσει ο χρήστης τότε βγαίνει μια λίστα με τα σκορ των παικτών που παίζουν το παιχνίδι. Τα σκορ χωρίζονται με τα καλύτερα σκορ όλων των εποχών, με σκορ αυτού του μήνα και σκορ αυτής της εβδομάδας. Επίσης υπάρχει και φόρμα εισόδου ώστε ένας παίκτης να κάνει λογαριασμό μέσω email ή facebook.

Εικονα 2: Βαθμολογίες παικτών

Αν ένας παίκτης θέλει να παίξει τυχαία με κάποιον άλλον τότε το παιχνίδι βρίσκει από μόνο του έναν ενεργό χρήστη και τον συνδέει μαζί του.

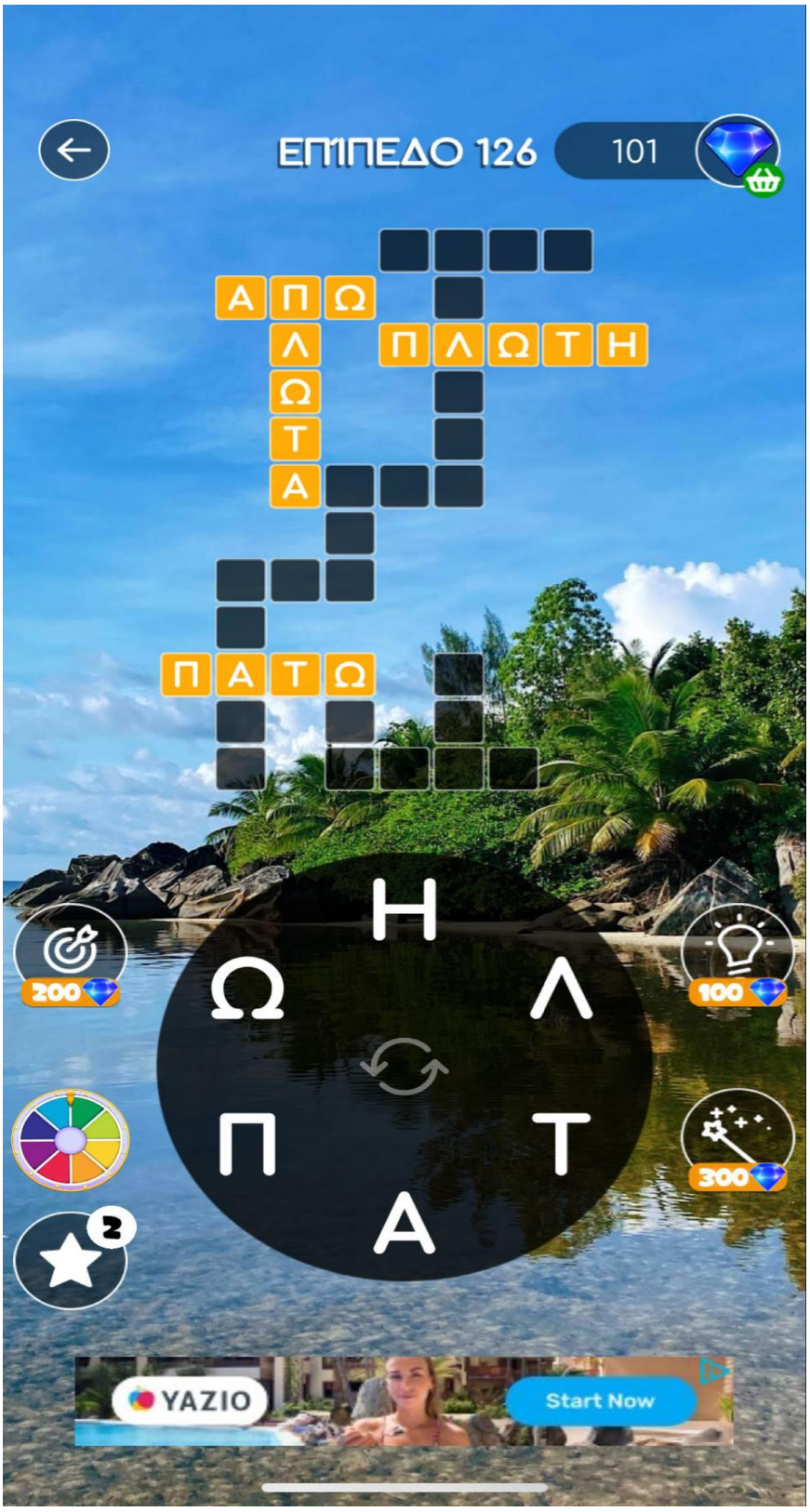


Εικόνα 3: Παιχνίδι Multiplayer

Αριστερά βλέπουμε τον χρήστη που συνδέεται με έναν τυχαίο χρήστη, και βλέπουμε και το σκορ δηλαδή πόσες ήττες και νίκες έχει στο παιχνίδι Multiplayer.

2.2 Επίπεδα & Βοήθειες Crossword Puzzle Free

Αφού ο χρήστης πατήσει στο επίπεδο τότε πηγαίνει στο ανάλογο επίπεδο και εμφανίζεται στην οθόνη του η δημιουργία του σταυρολέξου με την παλέτα γραμμάτων.

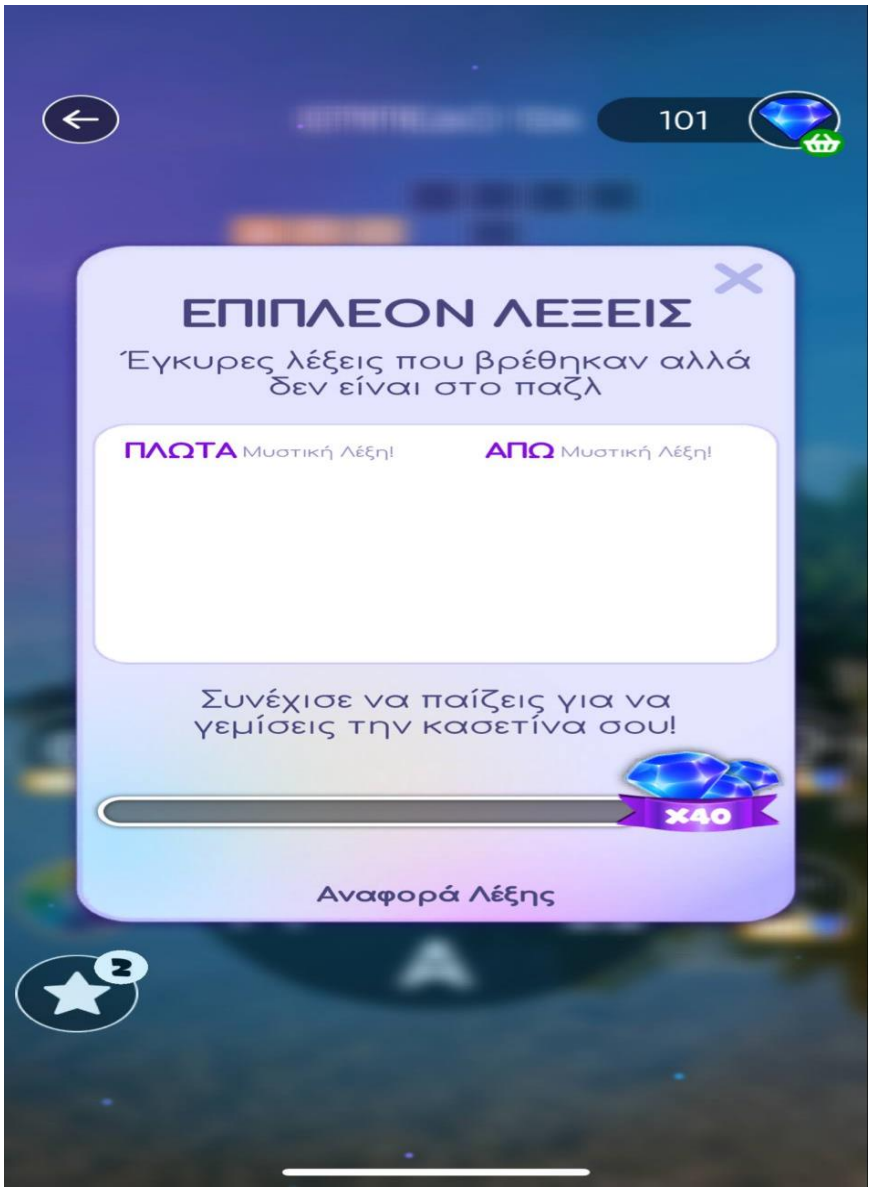


Εικόνα 4: Επίπεδο Crossword Puzzle Free

Στην οθόνη του ο χρήστης βλέπει τα ανάλογα γράμματα, το σκορ των διαμαντιών που έχει, θα εξηγήσουμε σε λίγο για αυτά, και τις βοήθειες που μπορεί να πάρει.

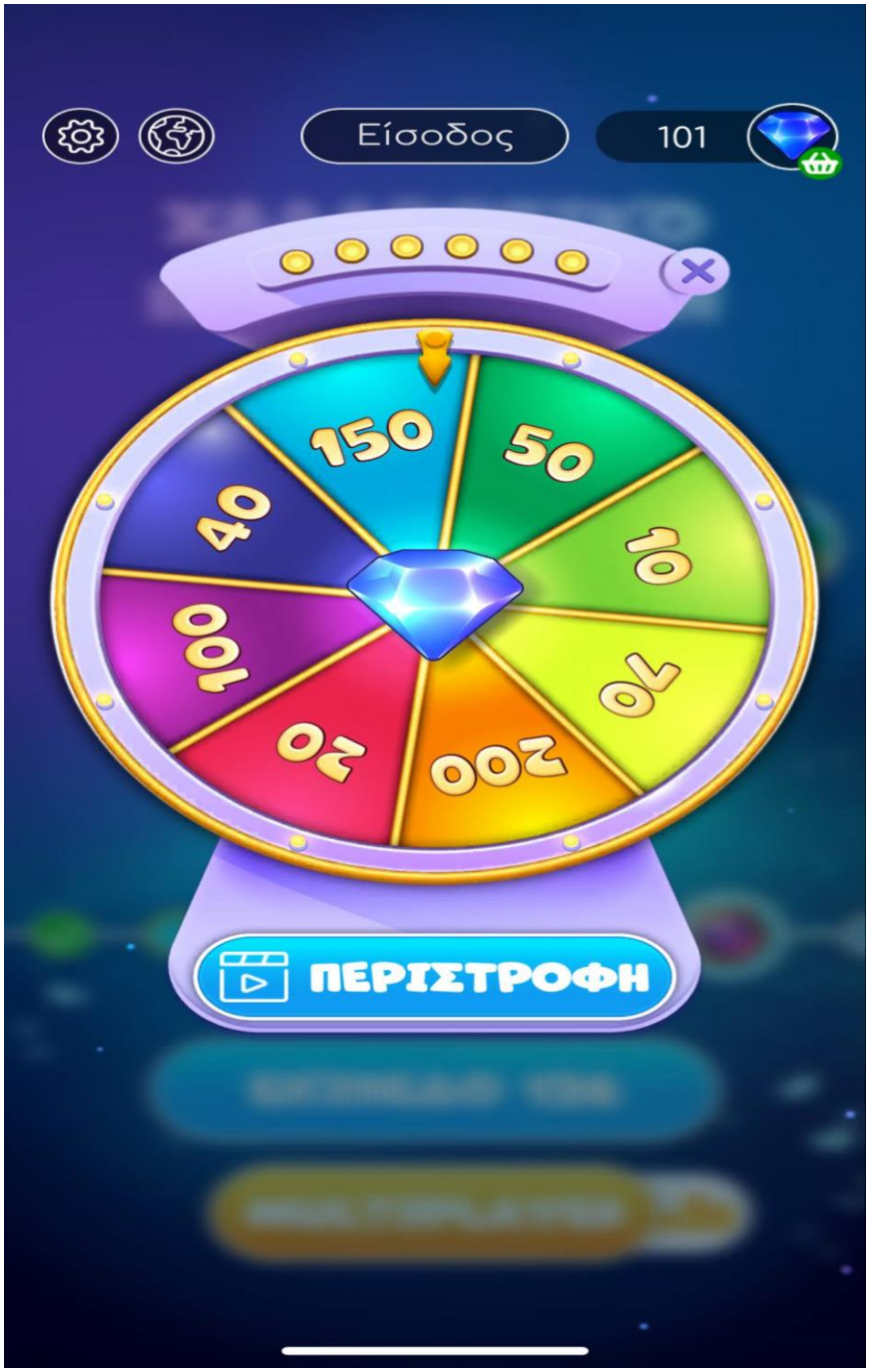
Υπάρχουν τρία είδη βοηθειών: Ο στόχος ο οποίος χρεώνει -200 διαμάντια τον χρήστη και του δίνει την δυνατότητα να διαλέξει ένα οποιοδήποτε κουτάκι που θέλει ο χρήστης ώστε να εμφανιστεί το γράμμα του. Η δεύτερη βοήθεια είναι το ραβδί το οποίο χρεώνει -300 διαμάντια και εμφανίζει μερικά τυχαία γράμματα μέσα στο σταυρόλεξο. Η τρίτη και τελευταία βοήθεια είναι η λάμπα η οποία αφαιρεί -100 διαμάντια και δίνει ένα γράμμα ως βοήθεια αλλά χωρίς να διαλέξει ο χρήστης ποιο θέλει να ανοίξει.

Κάτω αριστερά βλέπουμε ένα αστέρι εκεί μαζεύονται οι επιπλέον λέξεις, δηλαδή λέξεις που υπάρχουν στο λεξικό αλλά δεν είναι σαν λέξεις στο σταυρόλεξο, αφού μαζευτεί ένας αριθμός επιπλέον λέξεων το παιχνίδι αμοίβει τον χρήστη με έναν αριθμό διμαντιών που αλλάζει ανάλογα με το επίπεδο που προχωράει ο χρήστης.



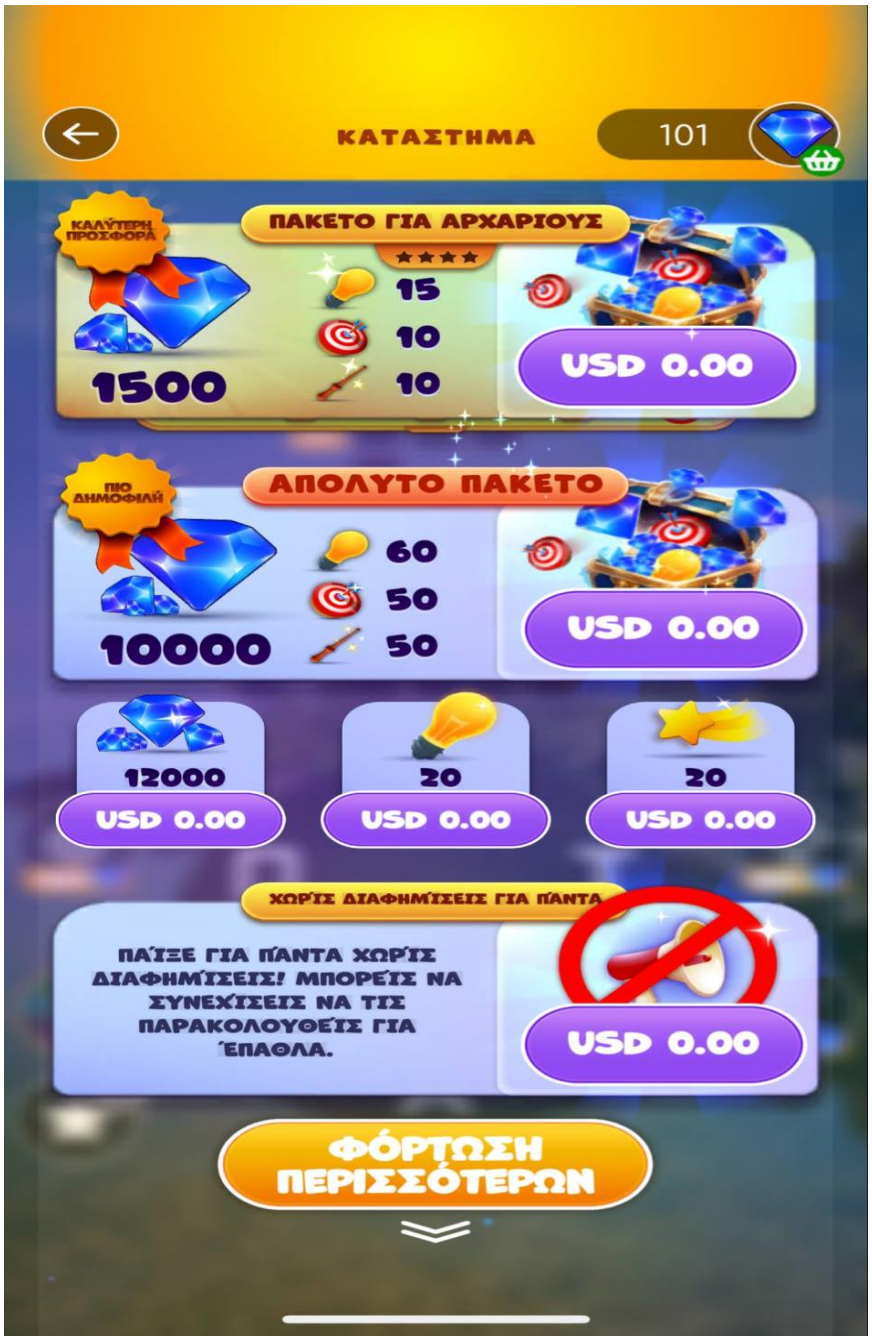
Εικόνα 5: Πίνακας Εξτρα λέξεων

Πάνω από το αστέρι υπάρχει ένας τροχός πολύχρωμος. Αυτός ο τροχός είναι για να δώσει έναν αριθμό διαμαντιών όταν ο χρήστης δεν έχει και χρειάζεται ώστε να χρησιμοποιήσει κάποια βοήθεια. Ο τροχός αυτος περιστρέφεται αφού πρώτα δείξει μία διαφήμιση στον χρήστη. Αφού τελειώσει η διαφήμιση τότε ο χρήστης παίρνει τα διαμάντια που θα του δώσει ο τροχός.



Εικόνα 6: Τροχός Διαμαντιών

Αν ο χρήστης χρειάζεται περισσότερα διαμάντια ώστε να πάρει βοήθειες πιο άνετα τότε μπορεί να κάνει αγορά αφού πληρώσει. Υπάρχει δηλαδή η δυνατότητα πληρωμής για βοήθειες.



Εικόνα 7: Πληρωμή για Βοήθειες.

2.3 Σύγκριση με Crossword Puzzle Free

Η έμπνευση της εφαρμογής που δημιουργήσαμε βασίζεται στο Crossword Puzzle Free, αλλά με πιο απλή μορφή. Στην υλοποίησή μας επιλέξαμε να διατηρήσουμε το παιχνίδι σε έναν πιο απλό, εστιασμένο πυρήνα, χωρίς να ενσωματώσουμε τα επιπλέον χαρακτηριστικά του “Crossword Puzzle Free” όπως:

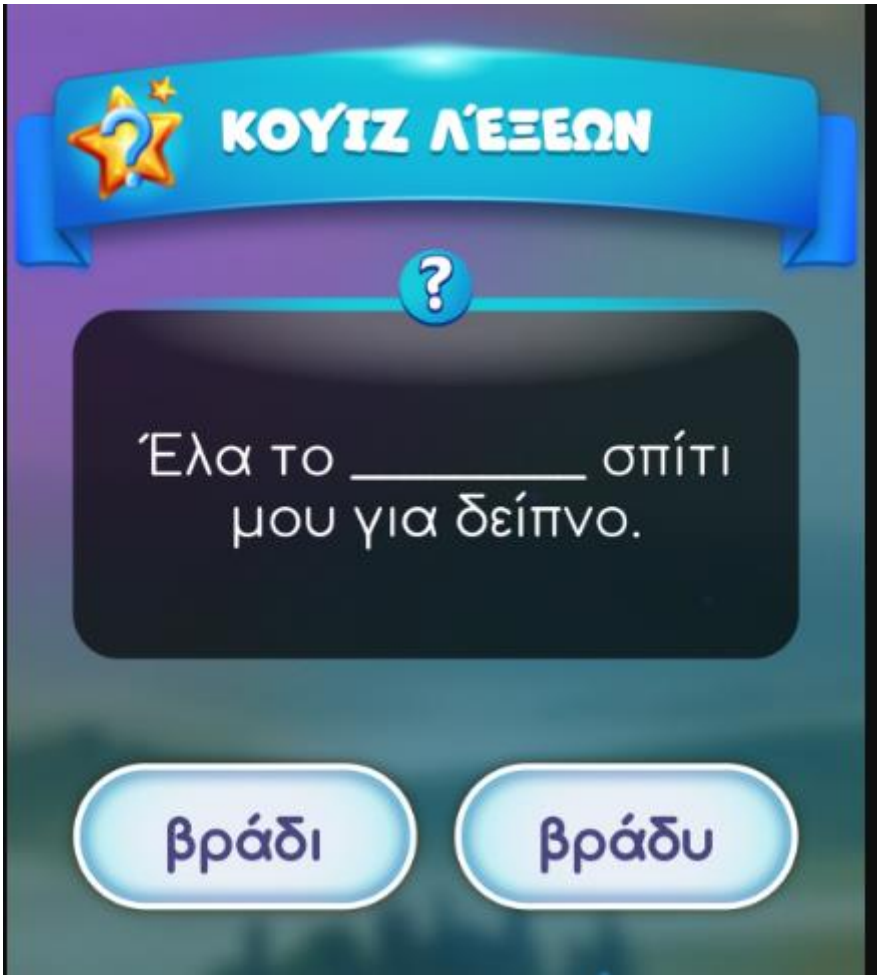
- **Multiplayer mode** και φόρμα εισόδου χρήστη
- **Τροχό τυχερών διαμαντιών** μετά από κάθε τρίτο επίπεδο
- **Εμφάνιση διαφημίσεων** και σύστημα πληρωμών (in-app purchases)

Αντίθετα, εμπνευστήκαμε από τον τρόπο με τον οποίο το original παιχνίδι αξιοποιεί την **παλέτα γραμμάτων** και τα **βοηθήματα** (hints, extra words), καθώς και από το clean-canvas UI του. Στη δική μας έκδοση αναπτύξαμε αυτές τις ιδέες με τις εξής παραλλαγές:

- **Παλέτα γραμμάτων**: Διατηρήσαμε τον κύκλο επιλογής γραμμάτων, αλλά προσθέσαμε δικό μας σύστημα ελέγχου ορθότητας και χρωματικής σήμανσης (κόκκινα highlights, timer).
- **Βοηθήματα**: Αντί για πολύπλοκα πακέτα hints ή διαφόρων ειδών power-ups, συμπεριλάβαμε τρία απλά hint με κόστος σε πόντους και διαμάντια, καθώς και ένα “extra word” bonus σε διαμάντια.

Επίσης το Crossword Puzzle Free δίνει εξτρα μπόνους επίπεδα που μπορείς να πάρεις διαμάντια. Αυτά τα επίπεδα είναι κάθε φόρα σε διαφορετική μορφή όπως να συμπληρωθεί η λέξη ή να βρούμε μία λέξη που καθώς εμφανίζονται γράμματα σε αυτήν τελειώνει ο χρόνος. Η δεύτερη εκδοχή έχει γίνει ακριβώς με τον ίδιο τρόπο και στην δικιά μας εφαρμογή.

Εικόνα 8 : Κουίζ λέξεων



Εικόνα 9: Συμπλήρωση λέξης

Η επιβράβευση αφού βρεθεί η λέξη είναι +45 διαμάντια για το μπόνους επίπεδο. Στην πρώτη εικόνα όπως είπαμε όσο περνάει ο χρόνος εμφανίζονται γράμματα της λέξης, αυτή η υλοποίηση έχει γίνει και στην εφαρμογή μας που θα παρουσιάσουμε παρακάτω με την μόνη διαφορά ότι αν δεν την βρει ο χρήστης και ο χρόνος τελειώσει τότε χάνει τα διαμάντια που θα έπαιρνε, αν δεν έχει διαμάντια τότε μειώνονται βαθμοί από το σκορ.

Κεφάλαιο 3ο: Περιγραφή του Project

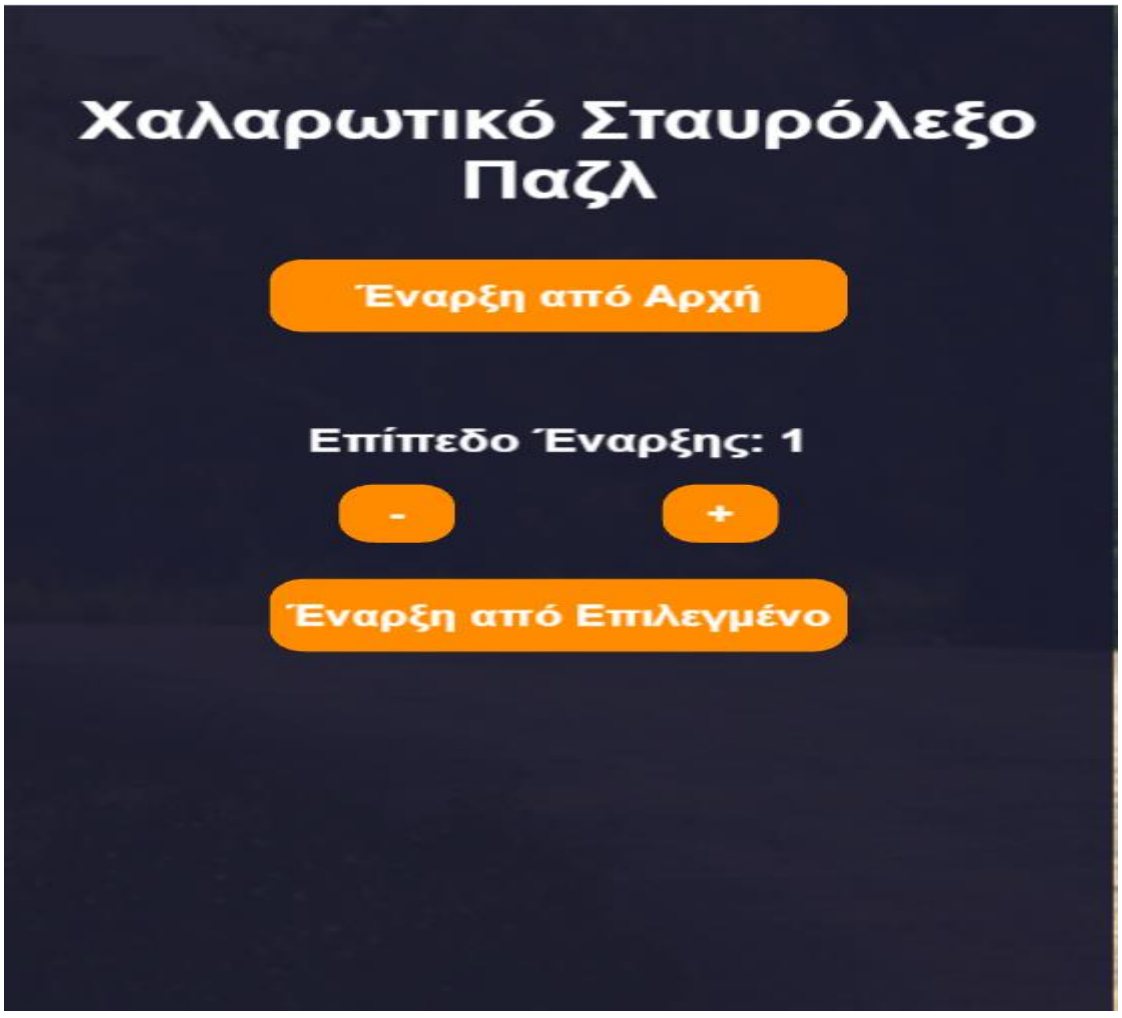
Το παρόν κεφάλαιο εστιάζει στην αναλυτική παρουσίαση του έργου που αναπτύχθηκε στο πλαίσιο της πτυχιακής εργασίας. Στις ενότητες που ακολουθούν, παρουσιάζονται αναλυτικά όλα τα συστατικά της εφαρμογής, από τη βασική ιδέα έως τα μηχανικά και διαδραστικά της μέρη.

3.1 Ιδέα και βασική λειτουργία

Η βασική ιδέα πίσω από την ανάπτυξη της εφαρμογής ήταν η δημιουργία ενός παιχνιδιού λέξεων σταυρόλεξου, το οποίο να προσφέρει ψυχαγωγία και ταυτόχρονα να ενισχύει το λεξιλόγιο του χρήστη. Η εφαρμογή υλοποιείται για περιβάλλον browser, χωρίς την ανάγκη εγκατάστασης, ώστε να είναι άμεσα προσβάσιμη από οποιαδήποτε συσκευή. Ο χρήστης καλείται να σχηματίσει λέξεις χρησιμοποιώντας ένα σύνολο τυχαίων γραμμάτων. Οι λέξεις αυτές συγκρίνονται με ένα λεξικό και αν είναι έγκυρες και συμπεριλαμβάνονται στο σταυρόλεξο του επιπέδου, αποκαλύπτονται στο πλέγμα σταυρόλεξου. Η εφαρμογή υποστηρίζει εκκίνηση μέσω αρχικού μενού, δίνοντας στον χρήστη τη δυνατότητα να ξεκινήσει νέο παιχνίδι ή να επιλέξει συγκεκριμένο επίπεδο. Αυτό ενισχύει την προσαρμοστικότητα της εμπειρίας και επιτρέπει την επανάληψη ή δοκιμή συγκεκριμένων σεναρίων χρήσης.

3.2 Συστατικά στοιχεία του gameplay

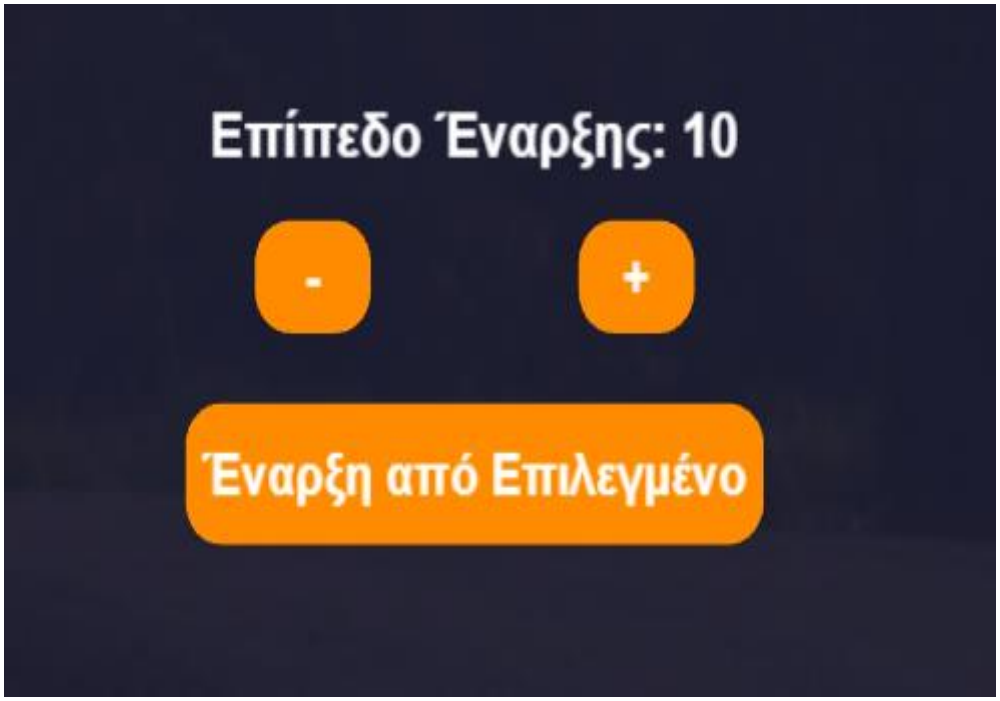
Το gameplay του παιχνιδιού αποτελείται από τα εξής βασικά στοιχεία, την παλέτα γραμμάτων στο κάτω μέρος της οθόνης, το σταυρόλεξο στο κέντρο, έναν μηχανισμό ελέγχου εγκυρότητας λέξεων και ενδείξεις για πόντους, διαμάντια, βοήθειες και επίπεδα. Παρακάτω βλέπουμε το αρχικό μενού του παιχνιδιού.



Εικόνα 10: Αρχικό μενού και επιλογές

Υπάρχουν δύο επιλογές στο αρχικό μενού για τον χρήστη. Η πρώτη είναι να ξεκινήσει από το πρώτο επίπεδο και να προχωράει στον βαθμό δυσκολίας καθώς περνάει τα επίπεδα ένα ένα. Ο βαθμός δυσκολίας αυξάνεται καθώς ο χρήστης φτάνει πιο μακριά, αυτό γίνεται λόγω των λέξεων που είναι περισσότερες και πιο μεγάλες με πιο πολλά γράμματα.

Η δεύτερη επιλογή είναι η "Έναρξη από επιλεγμένο" η οποία έχει δύο κουμπιά + και – ώστε ο χρήστης να μπορεί να αυξήσει και να μειώσει το επίπεδο δυσκολίας που θέλει να παίξει. Η ετικέτα «Επίπεδο Έναρξης» δείχνει τον αριθμό στον οποίο αυξάνεται η μειώνεται το επίπεδο επιλογής.



Εικόνα 11: Μενού επιλογής συγκεκριμένου επιπέδου-10

Στην επιλογή επιπέδου περιέχονται ως επιλογή και τα επίπεδα μπόνους που θα δουμε παρακάτω. Πατώντας την έναρξη ξεκινάμε από το καθορισμένο επίπεδο. Στην συγκεκριμένη περίπτωση ο χρήστης θα ξεκινήσει από το επίπεδο 10.

Αμέσως μετά την επιλογή ενός επιπέδου από την αρχική οθόνη, το παιχνίδι εκτελεί τα εξής βήματα:

Κλήση του LevelGenerator---Το LevelGenerator φορτώνει ή δημιουργεί τα δεδομένα για το επίπεδο: τη λίστα των γραμμάτων της παλέτας (letters) και τον πίνακα των λέξεων (words) μαζί με τις συντεταγμένες και την κατεύθυνσή τους. Αν πρόκειται για επίπεδο μπόνους, επιλέγει επίσης έναν «στόχο» που πρέπει να βρούμε.

Δημιουργία GridSection---Στο LevelScene καλείται η #initGridSection(), όπου:

- Υπολογίζονται τα ελάχιστα και μέγιστα row/column με βάση τις συντεταγμένες κάθε λέξης.
- Προσαρμόζεται δυναμικά το cellSize ώστε το πλέγμα να χωράει στην προκαθορισμένη περιοχή (προϊόν διαθέσιμου πλάτους/ύψους).

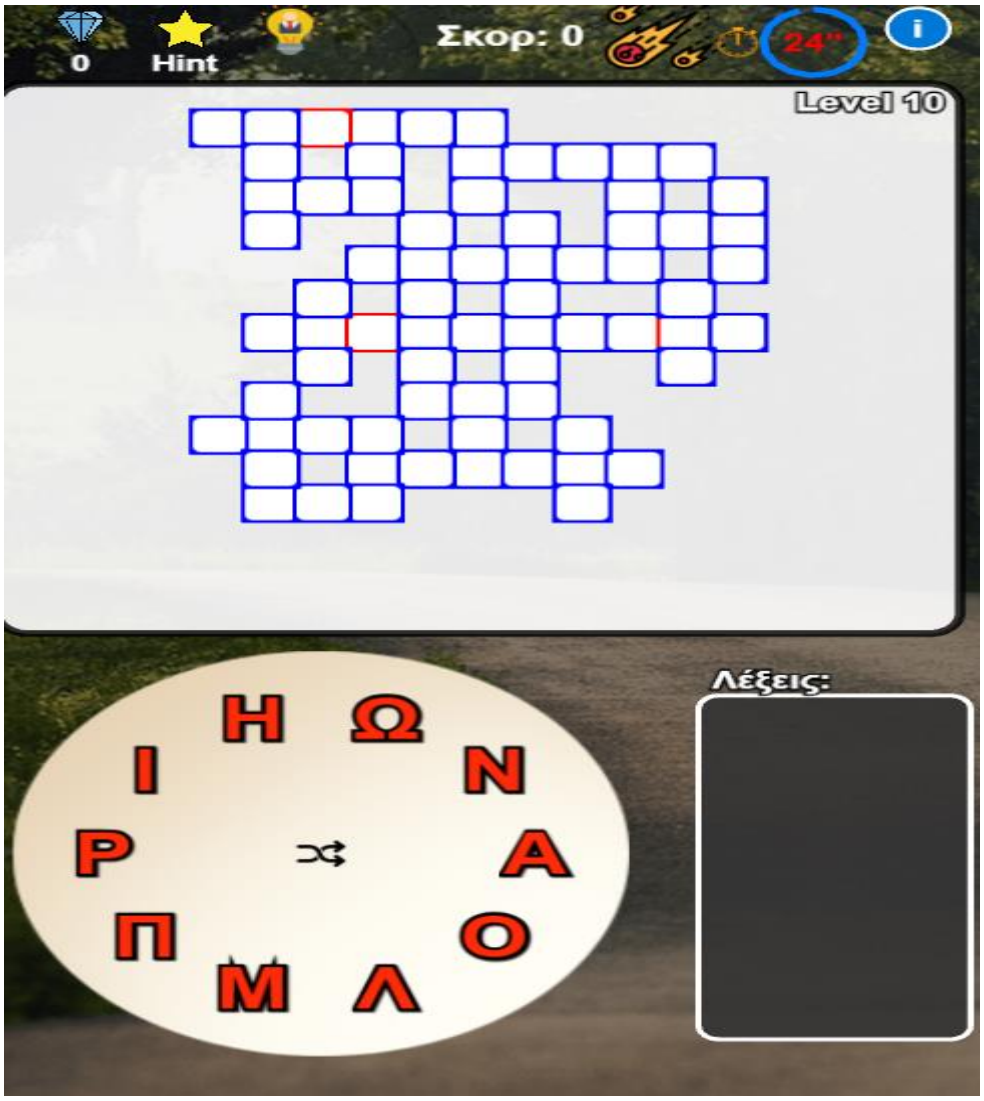
- Κάθε χαρακτήρας κάθε λέξης τοποθετείται μέσα σε ένα GridCell, που αρχικά έχει λευκό φόντο και μπλε περίγραμμα (locked state).

Εμφάνιση Παλέτας Γραμμάτων---Η LetterPalete τοποθετείται κάτω από το grid, «τραβάει» τα γράμματα levelLetters και περιμένει την αλληλεπίδραση: κάθε φορά που ο χρήστης σχηματίζει μια λέξη, η callback #gridControlCallBack ελέγχει αν ανήκει στο remaningLevelWords, αν είναι extra ή λεξικογενής.

UI και Έναρξη Παιχνιδιού---Ο LevelScene ολοκληρώνει την τοποθέτηση:

- Μπάρα σκορ και διαμαντιών πάνω αριστερά.
- Κουμπιά βοήθειας πάνω δεξιά.
- Πίνακας με τις ήδη βρεθείσες λέξεις δεξιά

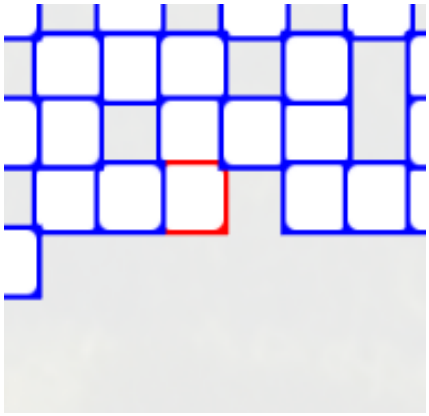
Μόλις εμφανιστούν το πλέγμα και η παλέτα, το επίπεδο ξεκινά. Η εικόνα που βλέπουμε είναι η εξής:



Εικόνα 12 : Εμφάνιση επιπέδου 10

Αφού το επίπεδο φορτωθεί και εμφανιστεί το πλέγμα, εκτελούνται τα ακόλουθα βήματα:

Highlight “special” κελιών---Αυθόρμητα, τυχαία επιλεγμένα 3 κελιά του grid αποκτούν έντονο κόκκινο περίγραμμα για 25″. Στο εσωτερικό τους αποθηκεύεται ένα εσωτερικό μετρητή («highlight bonus») που δηλώνει πόσα διαμάντια θα κερδίσεις αν βρεις τη λέξη τους μέσα στο χρόνο.



Εικόνα 13: Highlight κόκκινα κελιά

Εμφάνιση Circular Countdown Timer-Παράλληλα με τα κόκκινα περιγράμματα, στο πάνω-δεξιά μέρος της οθόνης εμφανίζεται ένας κυκλικός χρονομετρητής (ring) με εικονίδιο ⏱ και αριθμό (π.χ. “25″”). Καθώς περνάει ο χρόνος, ο δακτύλιος συρρικνώνεται ομαλά και ο αριθμός μειώνεται σε 24, 23, …, μέχρι το 0.

Εικόνα 14: Αντίστροφη μέτρηση

Μόλις φτάσει στο 0 το χρονόμετρο εξαφανίζεται και μαζί του και τα κόκκινα κελιά όπου δίνουν το έξτρα διαμάντι. Αν ο χρήστης βρει μία λέξη με κόκκινο κελί ενώ ο χρόνος έχει τελειώσει δεν αμοίβεται με έξτρα διαμάντι. Επίσης αν μία λέξη έχει παραπάνω από ένα κόκκινο κελί τότε ο χρήστης παίρνει έξτρα διαμάντια όσα είναι και τα κελιά που περιείχε η λέξη.

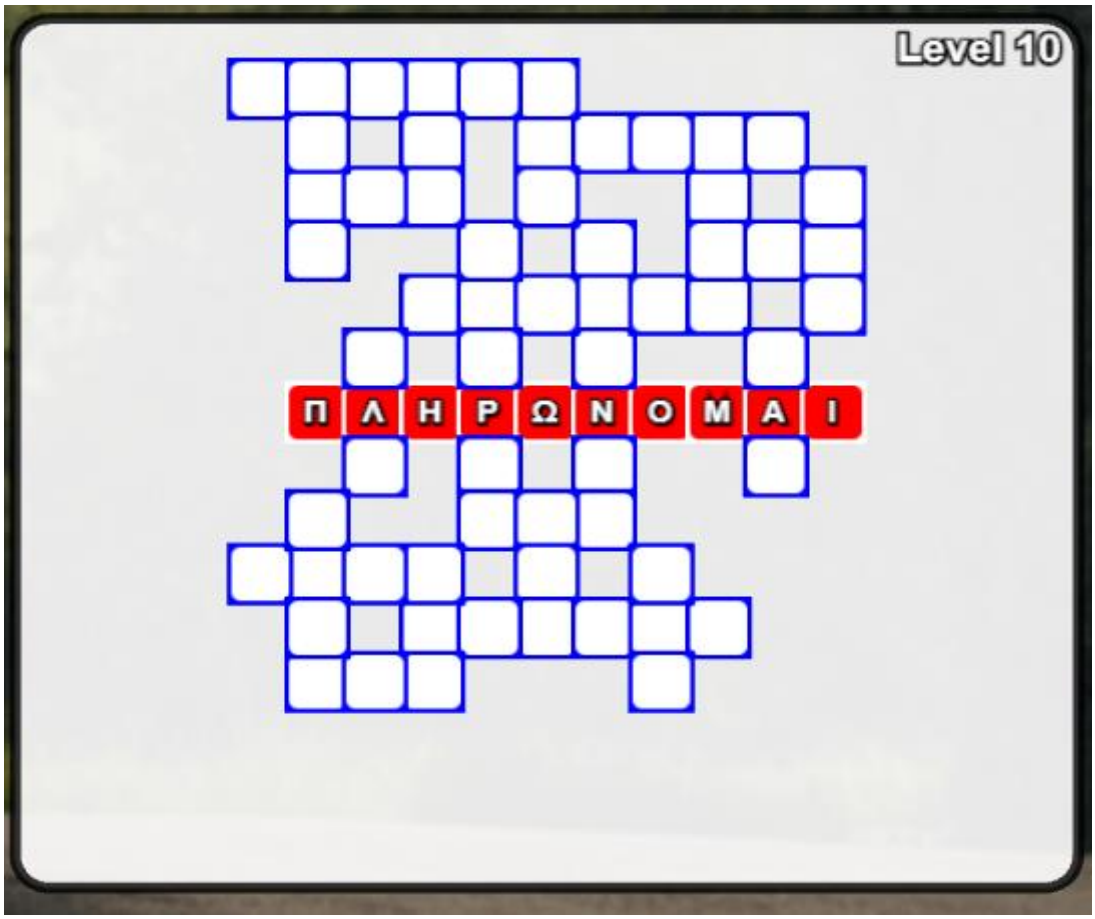
Ενεργοποίηση Bonus Διαμαντιών---Κατά τη διάρκεια των 25″, αν βρεις μια από τις λέξεις που περιέχουν κάποιο από τα κόκκινα κελιά, κερδίζεις τόσα διαμάντια όσα τα κελιά της λέξης που ήταν επισημασμένα. Το κουμπί «Diamond» στο πάνω μέρος αυξάνει αμέσως το απόθεμά σου και εμφανίζει popup «Κερδίσατε X διαμαντία!».

Εικόνα 15: Pop-up για έξτρα διαμάντι κόκκινου κελιού

Η πρώτη λέξη ώστε να δημιουργηθεί το σταυρόλεξο είναι η λέξη που χρησιμοποιεί όλα τα γράμματα της παλέτας. Αυτή η λέξη χρησιμοποιείται ως βάση ώστε να ενωθούν κι άλλες με τα γράμματα της βάσης και φτιαχτεί το σταυρόλεξο. Δηλαδή αφού φορτωθεί το λεξικό μας με όλες τις έγκυρες λέξεις, υπολογίζεται ανάλογα με το επίπεδο δυσκολίας, πόσα μοναδικά γράμματα χρειαζόμαστε.

Αφού σχηματίσουμε την λέξη με τα γραμμάτα της παλέτας η λέξη μπαίνει στο σταυρόλεξο συμπληρώνοντας τα κατάλληλα κουτάκια και μπαίνοντας στον πίνακα δεξία με τις λέξεις που βρήκε ο χρήστης.

Εικόνα 16: Γράμματα παλέτας και πίνακας λέξεων



Εικόνα 17: Συμπλήρωση λέξης στο σταυρόλεξο

Αυτή η λέξη που χρησιμοποιείται ως βάση ονομάζεται σπόρος (pangram). Τα γράμματα της λέξης σπόρου ανακατεύονται και γίνονται η τελική σειρά γραμμάτων που θα εμφανιστεί στην παλέτα γραμμάτων. Με βάση τα γράμματα της παλέτας, τρέχει η συνάρτηση generateWords που φιλτράρει το λεξικό για όλες τις λέξεις (δίχως να υπερβαίνουν κάποιο μέγιστο μήκος) που μπορούν να σχηματιστούν αποκλειστικά με αυτά τα γράμματα. Η ίδια η pangram λέξη αφαιρείται από αυτή τη λίστα **(για να μη διπλομετρηθεί).**

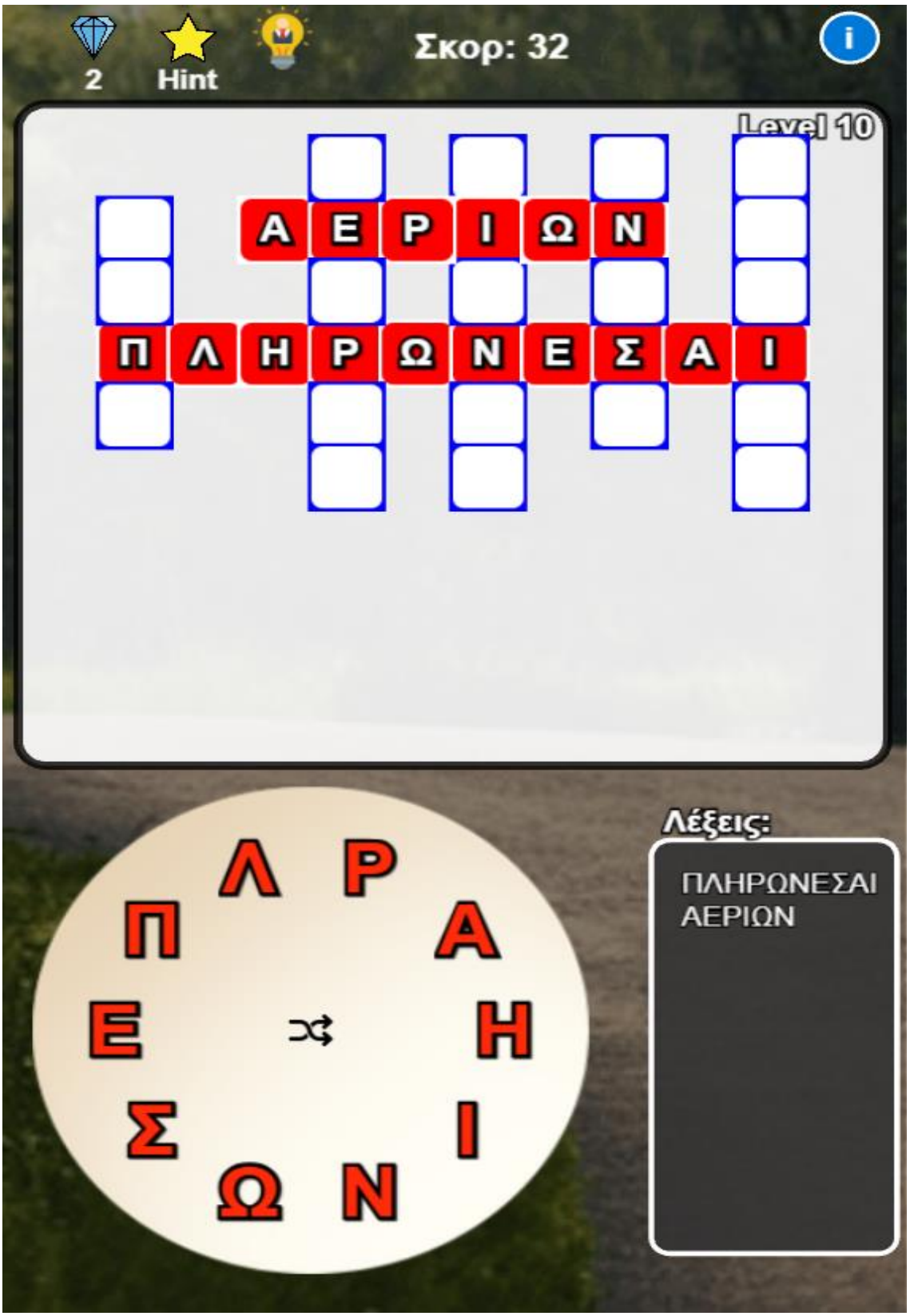
Με τη λίστα όλων των επιτρεπόμενων λέξεων (το pangram + τις άλλες valid), καλείται η generateCrossword. Αυτή τοποθετεί όσες περισσότερες λέξεις μπορεί σε ένα πλέγμα (με διασταύρωση γραμμών/στηλών), εξασφαλίζοντας ότι το pangram θα εμφανιστεί. Η έξοδος είναι ένα array από tuples [x, y, λέξη, κατεύθυνση] (H ή V).

Το LevelScene παίρνει το array [x,y,λέξη,dir] και δημιουργεί έναν προσωρινό χάρτη (cellMatrix) όπου κάθε θέση (row,col) χαρτογραφεί ένα αντικείμενο με: letter, το γράμμα που βρίσκεται εκεί, και words, τη λίστα των λέξεων που περνούν από το κελί (για σταυρώσεις). Υπολογίζονται τα minRow, maxRow, minCol, maxCol ώστε να ξέρουμε πόσες γραμμές/στήλες έχει το πλέγμα.

Με βάση το διαθέσιμο πλάτος/ύψος στην οθόνη, υπολογίζεται το cellSize ώστε όλα τα κελιά να χωράνε αναλογικά και να παραμένουν αναγνώσιμα.

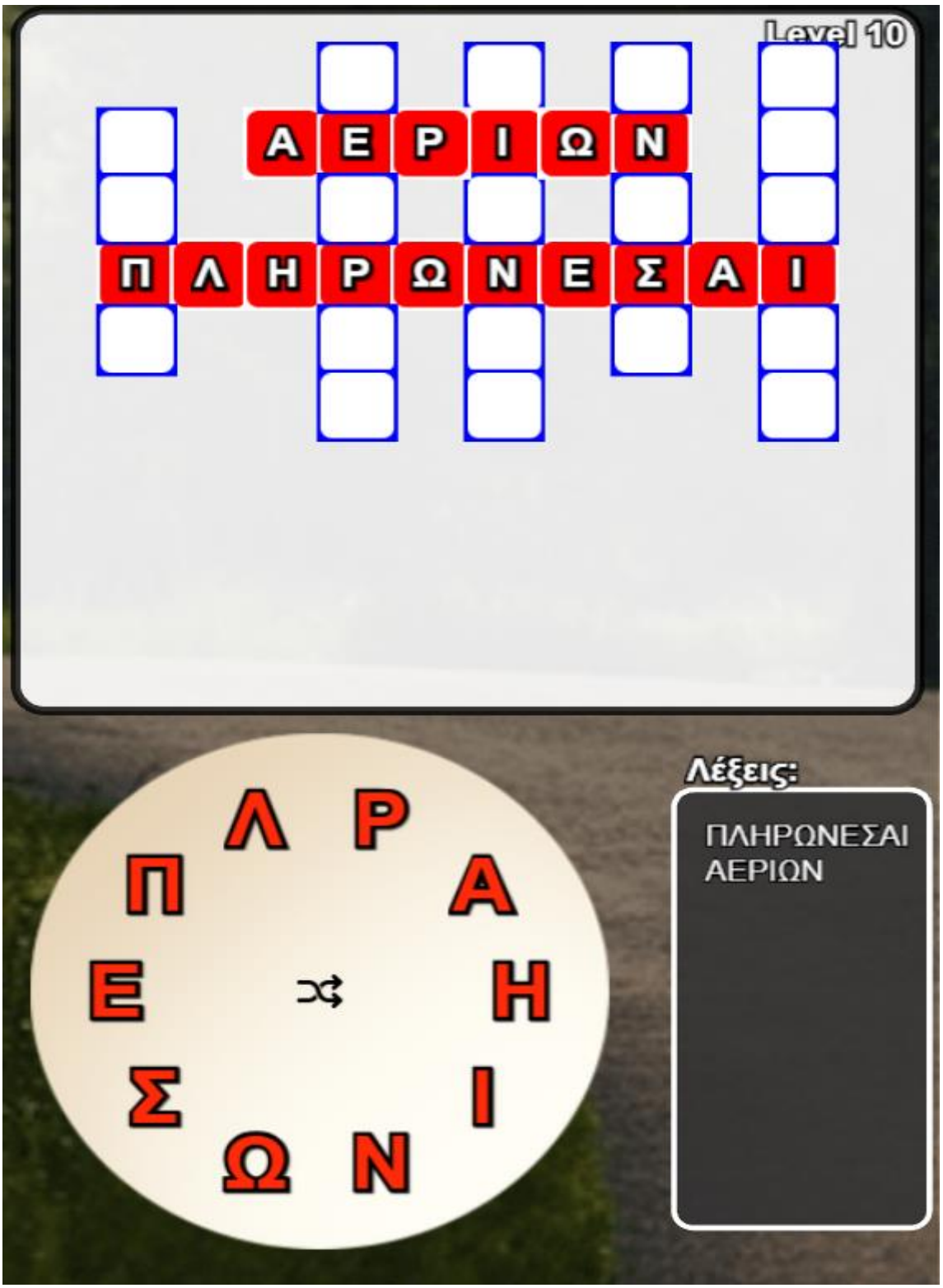
Εικονα 18: Σχηματισμός λέξης με σύρσιμο γραμμάτων

Ο παίκτης σέρνει τα γράμματα της παλέτας για να σχηματίσει τη λέξη. Μόλις ολοκληρώσει τα γράμματα, απελευθερώνει, και τα αντίστοιχα κελιά στο grid «ξεκλειδώνουν» (αλλάζουν χρώμα).



Εικόνα 19: Συμπλήρωση λέξεων αύξηση σκορ

Αμέσως μετά, ο παίκτης επιλέγει τα γράμματα για την επόμενη λέξη. Με το που την επιβεβαιώνει η λέξη προστίθεται στην θέση της μέσα στο σταυρόλεξο και στον πίνακα λέξεων κάτω από την προηγούμενη λέξη. Το σκορ αυξάνεται κατά 2 πόντους ανά γράμμα, αν δηλαδή η λέξη έχει 7 γράμματα τότε το σκορ θα αυξηθεί κατά 2×7 = +14 πόντους.

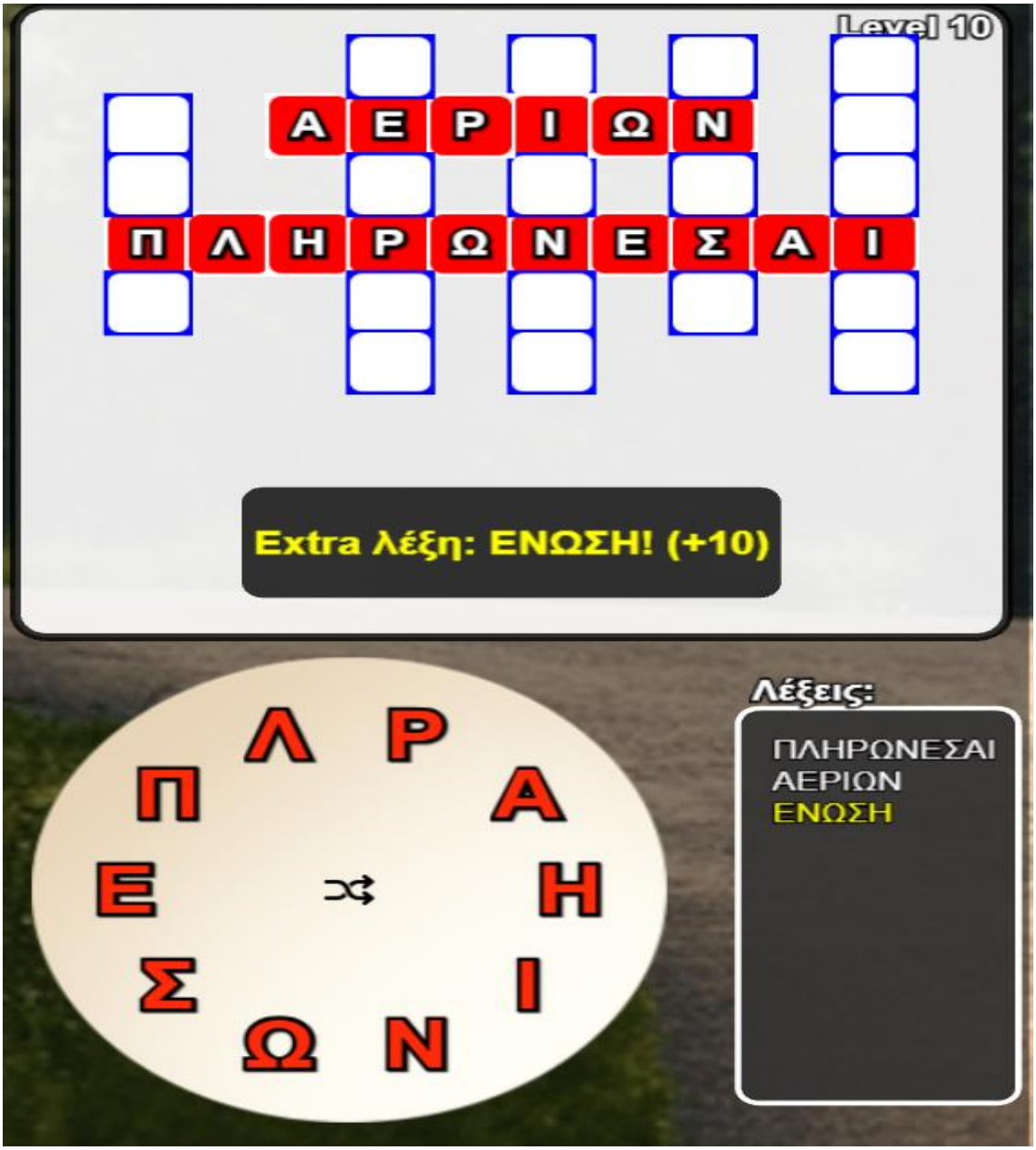


Εικονα 20: Προσθήκη λέξης ΠΛΗΡΩΝΕΣΑΙ

Με την λέξη ΠΛΗΡΩΝΕΣΑΙ το σκορ αυξήθηκε κατά 20 πόντους γιατί όπως είπαμε παιρνει δύο πόντους ο χρήστης για κάθε γράμμα. Και όπως φαίνεται στον πίνακα η λεξη πήγε στο κάτω μέρος από την προηγούμενη.

3.3 Εντοπισμός & Βαθμολόγηση Extra Λέξεων

Εκτός από τις κύριες λέξεις που αποτελούν το κύριο σώμα του σταυρόλεξου, το παιχνίδι υποστηρίζει και **extra λέξεις**—λέξεις που μπορούν να σχηματιστούν από τα ίδια γράμματα της παλέτας αλλά δεν μπήκαν στο πλέγμα ως διασταυρωμένες.



Εικονα 21: Extra λέξη και πόντοι

Κατά το στάδιο δημιουργίας επιπέδου, αφού επιλεγεί η pangram λέξη και εντοπιστούν οι κύριες λέξεις, ο αλγόριθμος συγκεντρώνει όλες τις εναπομένουσες λέξεις που μπορούν να σχηματιστούν από την παλέτα (generateWords). Οι λέξεις αυτές που δεν τοποθετήθηκαν στο grid χαρακτηρίζονται "extraWords" και επιστρέφονται στον LevelScene.Στο παραπάνω παράδειγμα φαίνεται ότι ο χρήστης έχει βρει μία έξτρα λέξη και εμφανίζεται το συγκεκριμένο pop-up.

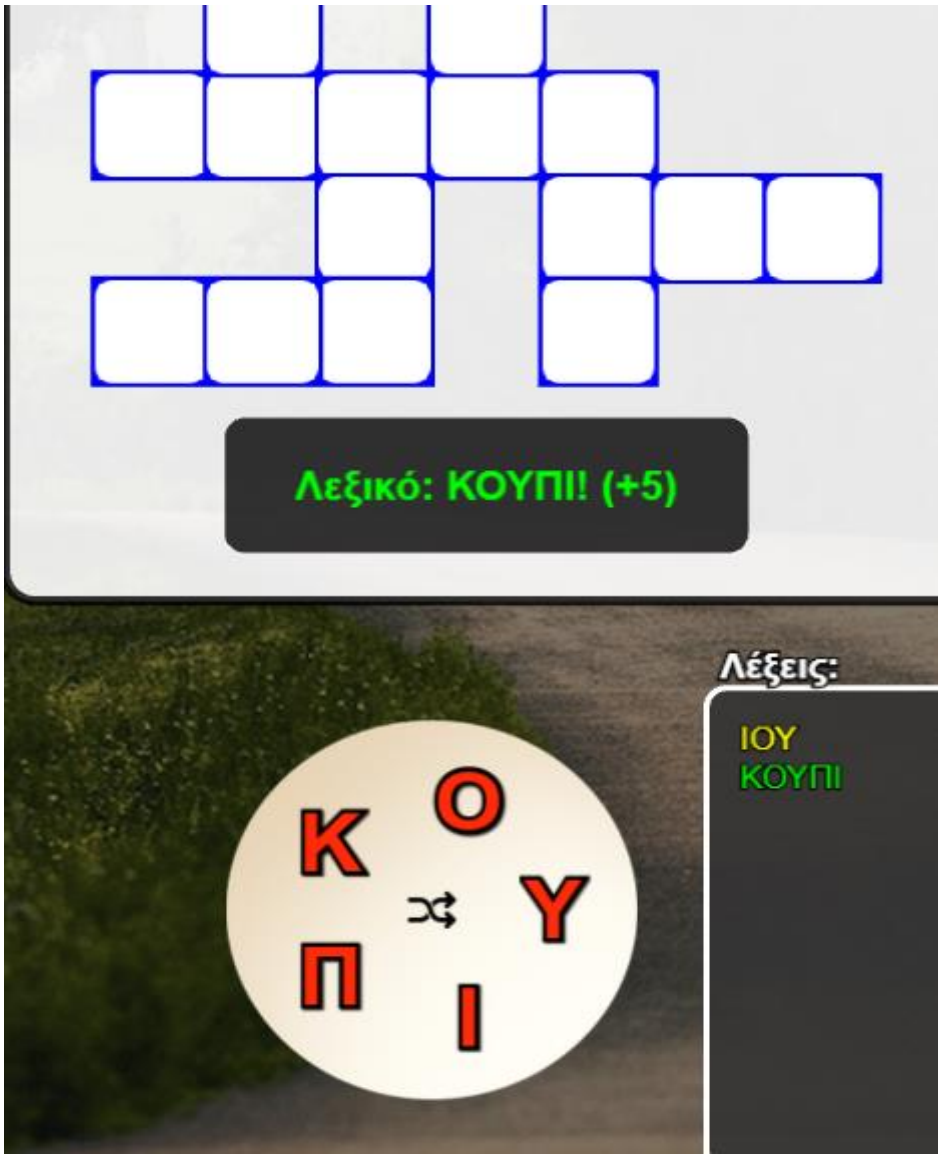
Η LetterPalete εμφανίζεται κάτω από το grid, παίρνει τα levelLetters και, όταν ο χρήστης στέλνει μια λέξη, τρέχει η callback #gridControlCallBack. Αν είναι στο levelWords: ξεκλειδώνει τα κατάλληλα κελιά, αφαιρεί τη λέξη από τη λίστα remaningLevelWords και τρέχει onWordFound. Αν είναι στις extraWords: τρέχει onExtraWordFound. Αν είναι στο ευρύ λεξικό: τρέχει onDictionaryWordFound.

Οι extra λέξεις δεν εμφανίζονται ως κουτιά στο πλέγμα, αλλά μπορούν να βρεθούν μέσω της παλέτας γραμμάτων ακριβώς όπως και οι κύριες. Αν ο παίκτης σχηματίσει μια τέτοια λέξη (που υπάρχει στη λίστα extraWords), το παιχνίδι την χρωματίζει με κίτρινο χρώμα, όπως φαίνεται και στον πίνακα λέξεων δεξιά. Όσον αναφορά την βαθμολόγηση κάθε extra λέξη αποδίδει **2 πόντους ανά γράμμα** (όπως οι κύριες), π.χ. μια 5-γράμματη extra λέξη δίνει +10 πόντους στο σκορ. Η διαφορά είναι ότι δεν ξεκλειδώνει νέα κελιά στο πλέγμα—είναι ένα bonus scoring mechanism.

Επειδή οι extra λέξεις μπορεί να είναι σύντομες ή ασύνδετες με τον κύριο σχεδιασμό, σε βοηθούν να μαζεύεις επιπλέον πόντους χωρίς να χρειάζεται να "σηκώνεις" ολόκληρο το πλέγμα. Ιδανικό για όσους στοχεύουν σε υψηλό σκορ ή για να καλύψουν "κενά" όταν κολλάνε στη λύση του σταυρόλεξου.

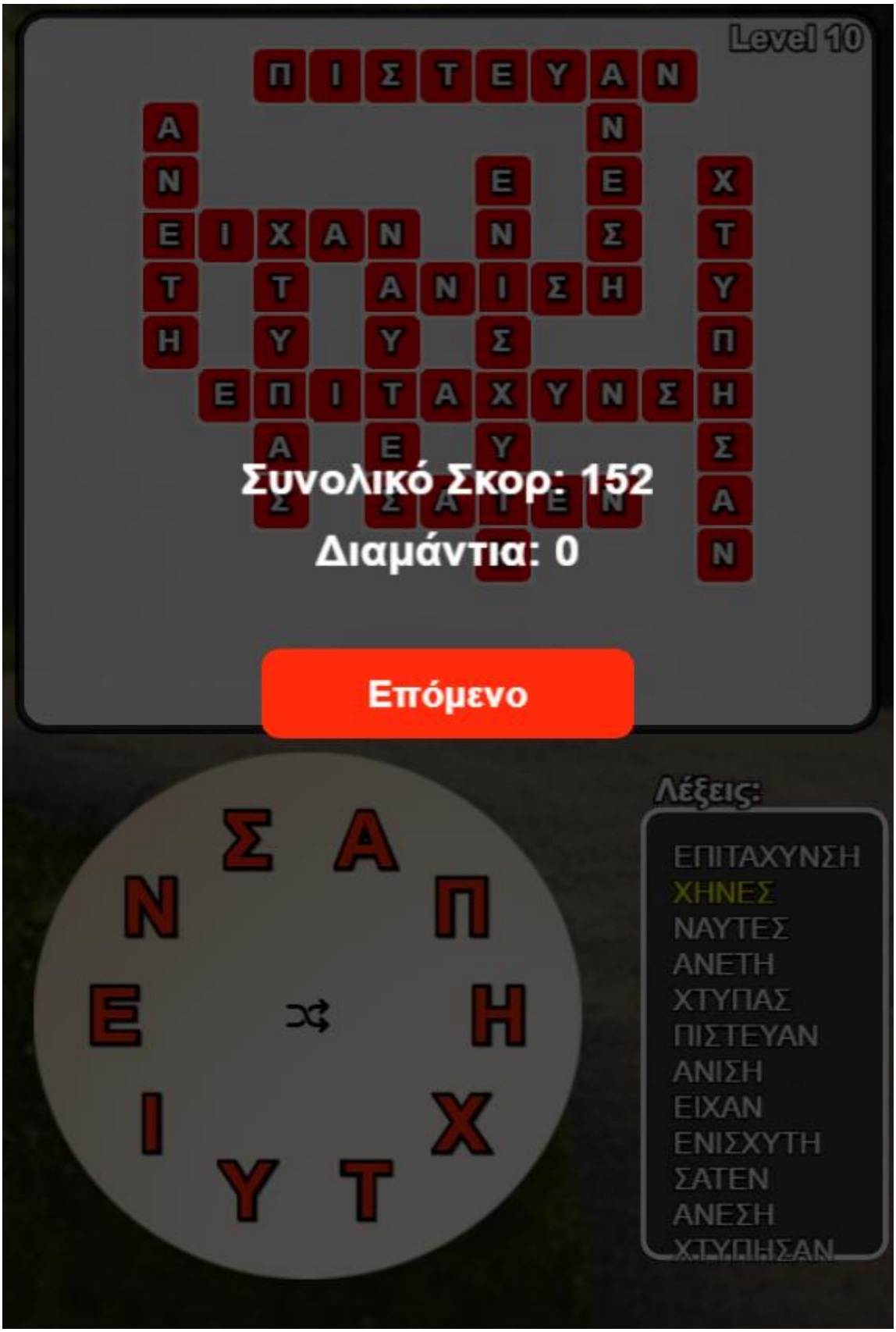
Επίσης είναι ένας τρόπος ώστε να αυξηθεί το σκορ και να μπορεί ο χρήστης να χρησιμοποιήσει κάποια από τις βοήθειες του παιχνιδιού. Πέρα από τις κύριες λέξεις του σταυρόλεξου και τις extra λέξεις, το παιχνίδι αναγνωρίζει και λέξεις λεξικού—λέξεις που δεν εμφανίζονται στο grid ή στη λίστα των extra, αλλά υπάρχουν στο ευρύτερο λεξικό της εφαρμογής.

Οι λέξεις λεξικού προστίθενται στη δεξιά λίστα **με πράσινο χρώμα** για να ξεχωρίζουν. Αυτό δείχνει σαφώς στον παίκτη ότι βρήκε μια "επιπλέον" λέξη που δεν είχε προβλεφθεί ούτε στο σταυρόλεξο ούτε στα extra. Για κάθε γράμμα τέτοιας λέξης δίνεται **1 πόντος** (χαμηλότερο ποσοστό, καθώς πρόκειται για απλό λεξικό). Παράδειγμα: η λέξη "ΚΟΥΠΙ" (5 γράμματα) δίνει +5 πόντους. Εμφανίζεται popup "Λεξικό: ΚΟΥΠΙ! (+5)" με πράσινο φόντο.



Εικόνα 22: Λέξη Λεξικού

Μετά από κάθε εύρεση λέξης (κύρια, extra ή λεξικού), ο κώδικας ελέγχει αν η λίστα remaningLevelWords έχει αδειάσει. Στην πράξη σημαίνει: αν δεν απομένει καμία λέξη του σταυρόλεξου που δεν έχει βρεθεί, τότε το επίπεδο ολοκληρώνεται. Μόλις εντοπιστεί η ολοκλήρωση, εμφανίζεται ένα popup στην οθόνη με μήνυμα “Επίπεδο Ολοκληρώθηκε!” ή “Congratulations!”, ανάλογα με το localization.



Εικόνα 23: Γέμισμα σταυρολέξου

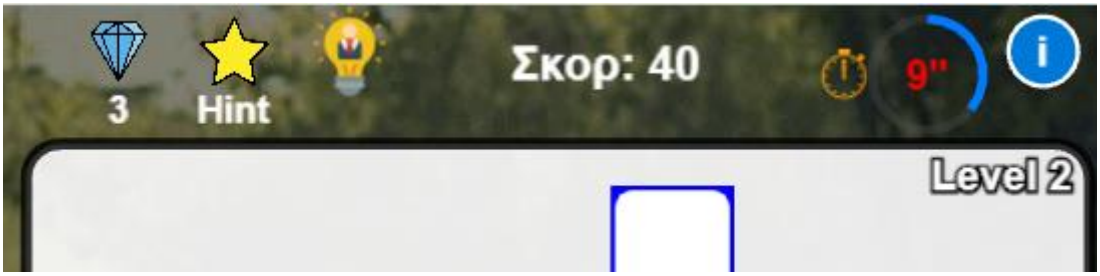
Εικόνα 24: Κουμπί επόμενου επιπέδου

Πατώντας το κουμπί Επόμενο προχωράμε στο επόμενο επίπεδο. Το περιβάλλον επαναφορτώνει νέο LevelScene με διαφορετικά γράμματα, νέες λέξεις, και ανανεωμένη παλέτα. Το συνολικό σκορ και τα διαμάντια του παίκτη αποθηκεύονται στο session (ή στο localStorage) ώστε, σε περίπτωση ανανέωσης της σελίδας ή προσωρινού κλεισίματος, να συνεχίσει από εκεί που σταμάτησε. Αν γίνει reload, το Game διαβάζει τις τιμές από τον αποθηκευτικό χώρο και τις περνά στο νέο LevelScene.

3.4 Βοηθήματα & Χαρακτηριστικά

Το παιχνίδι περιλαμβάνει ένα πλούσιο σύστημα βοηθημάτων, σχεδιασμένο ώστε να προσφέρει στον παίκτη επιλογές όταν "κολλήσει" ή απλώς θέλει να βελτιώσει το σκορ του. Κάθε τύπος βοήθειας έχει διαφορετικό σκοπό, κόστος και αποτέλεσμα – από το ξεκλείδωμα μεμονωμένων γραμμάτων ως την άμεση λύση του σταυρόλεξου. Στα επόμενα κεφάλαια θα αναλύσουμε αναλυτικά:

- Τι κάνει κάθε βοήθεια
- Πώς ενεργοποιείται στην πράξη
- Ποιο είναι το κόστος της (πόντοι ή διαμάντια)
- Ποια στρατηγική ακολουθούμε για βέλτιστη χρήση



Εικόνα 25: Βοήθειες και χρήση τους

3.5 Κουμπί Πληροφορίας

Στην επάνω δεξιά γωνία, δίπλα από τον χρονομετρητή, βρίσκεται το κουμπί **i** (Info). Αυτό εξυπηρετεί άμεση πρόσβαση στους κανόνες. Πατώντας το, ανοίγει ένα overlay με όλα τα βασικά σημεία του παιχνιδιού:

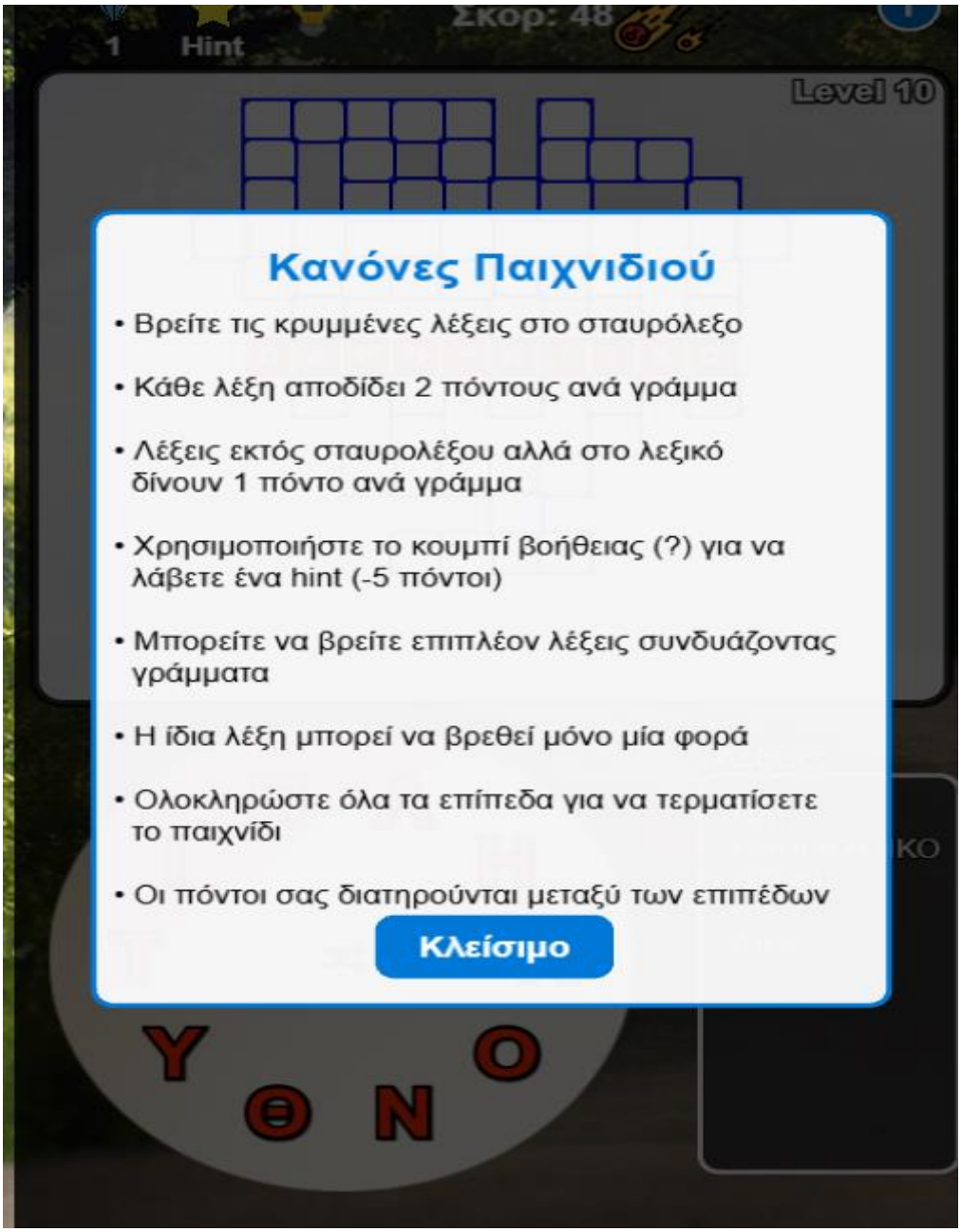
1)Σκοπός και ροή (βρες όλες τις λέξεις στο σταυρόλεξο)

2)Πώς βαθμολογούνται οι κύριες λέξεις (2 πόντοι ανά γράμμα)

3)Extra λέξεις (2 πόντοι/γράμμα) και λεξικογενείς λέξεις (1 πόντος/γράμμα)

4)Λειτουργίες βοήθειας (★ Hint, Help, Comet, Λάμπα) με κόστος και περιορισμούς,

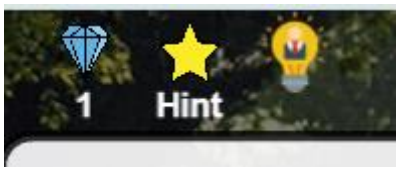
5)Τρόπος μετάβασης επιπέδου μόλις βρεθούν όλες οι λέξεις



Εικόνα 26: Κανόνες παιχνιδιού pop-up

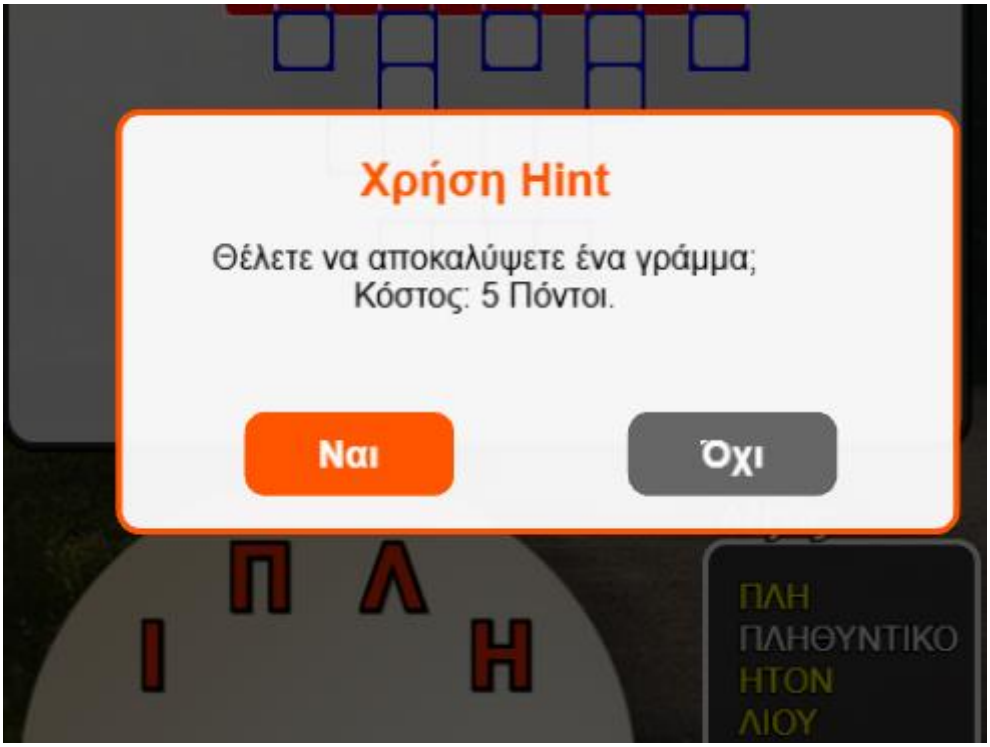
3.6 Αποκάλυψη Γράμματος (Hint)

Το ★ Hint είναι η πιο γρήγορη και άμεση βοήθεια όταν σε χωρίζουν ελάχιστα γράμματα από τη λύση μιας λέξης. Σκοπός είναι να ξεκλειδώσει ένα μόνο γράμμα μέσα στο σταυρόλεξο, ώστε να σε βοηθήσει να συμπληρώσεις πιο εύκολα τη λέξη που έχεις επιλέξει.Η ενεργοποίηση τους γίνεται πατώντας το αστέρι που βρίσκεται ακριβώς δίπλα στο εικονίδιο των διαμαντιών στην κορυφή της οθόνης.



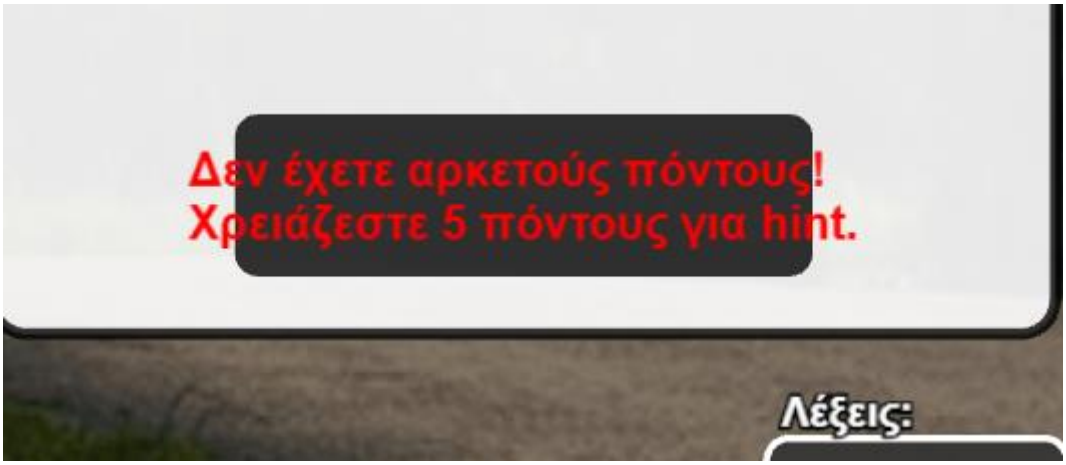
Εικόνα 27: Hint βοήθεια

Μόλις πατήσουμε το αστέρι για να χρησιμοποιήσουμε την βοήθεια εμφανίζεται ένα μικρό παράθυρο επιβεβαίωσης: “Θέλετε να αποκαλύψετε ένα γράμμα; Κόστος 5 πόντοι.” Και μπορεί ο χρήστης να επιλέξει ΝΑΙ ή ΟΧΙ.



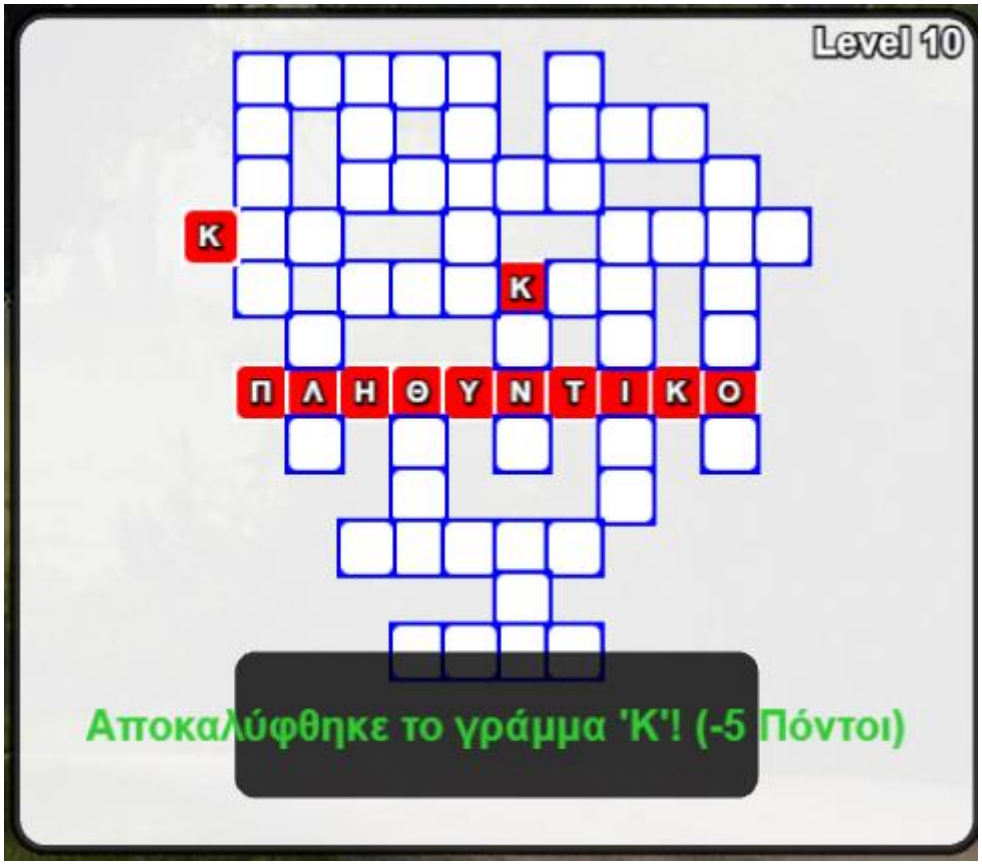
Εικόνα 28: Pop-up Hint

Αν επιλέξει ο χρήστης ΟΧΙ τότε το παράθυρο κλείνει χωρίς να γίνει κάτι. Αν πατήσει ΝΑΙ τότε η βοήθεια του αστεριού αφαιρεί **5 πόντους** από το συνολικό σου σκορ. Αν δεν διαθέτει 5 πόντους, εμφανίζεται σχετικό μήνυμα και δεν ενεργοποιείται η βοήθεια.



Εικόνα 29: Ενημέρωση για πόντους

Μετά την χρήση της βοήθειας εμφανίζεται τυχαίο γράμμα και μήνυμα το οποίο ενημερώνει τον χρήστη.



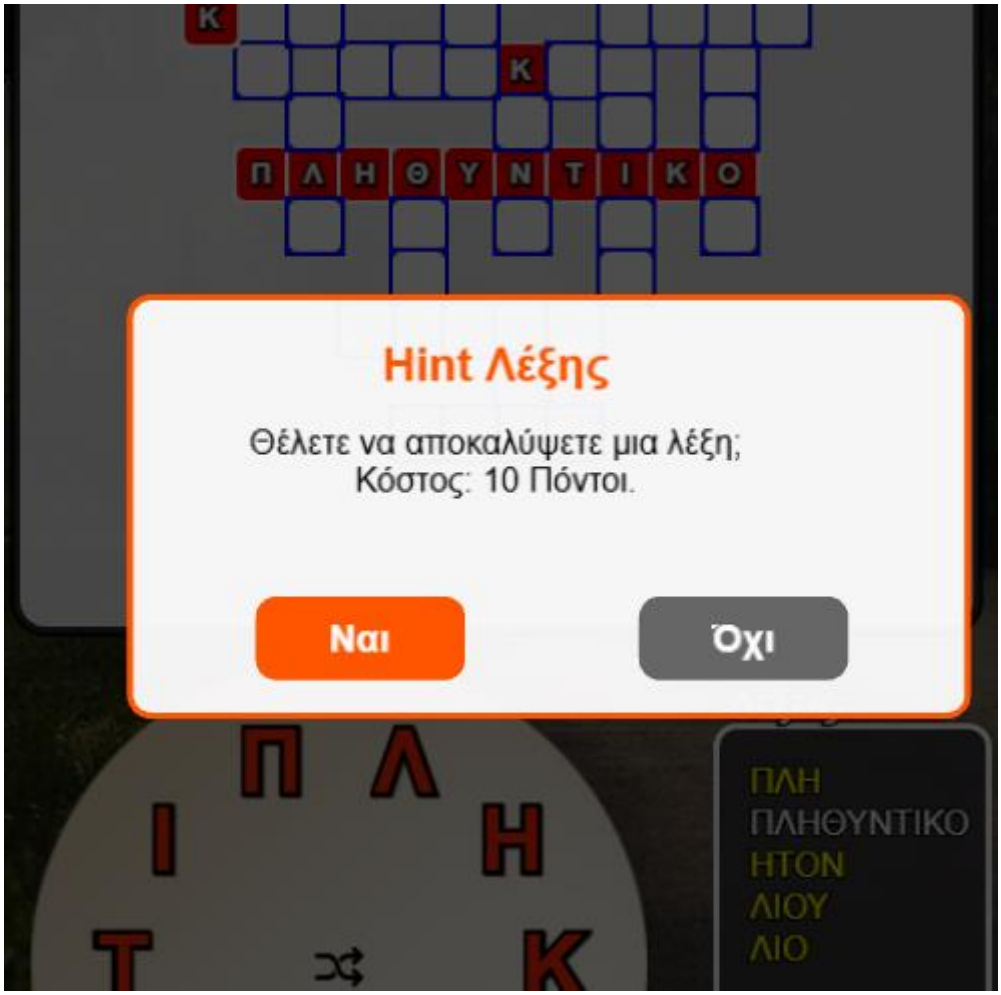
Εικόνα 30: Παράθυρο ενημέρωσης χρήστη

Αν αυτό το γράμμα συμπληρώνει πλήρως μια λέξη, η λέξη καταχωρείται αμέσως ως “βρεθείσα” και λαμβάνεις τους αντίστοιχους πόντους (2 × μήκος λέξης).

3.7 Αποκάλυψη Ολόκληρης Λέξης

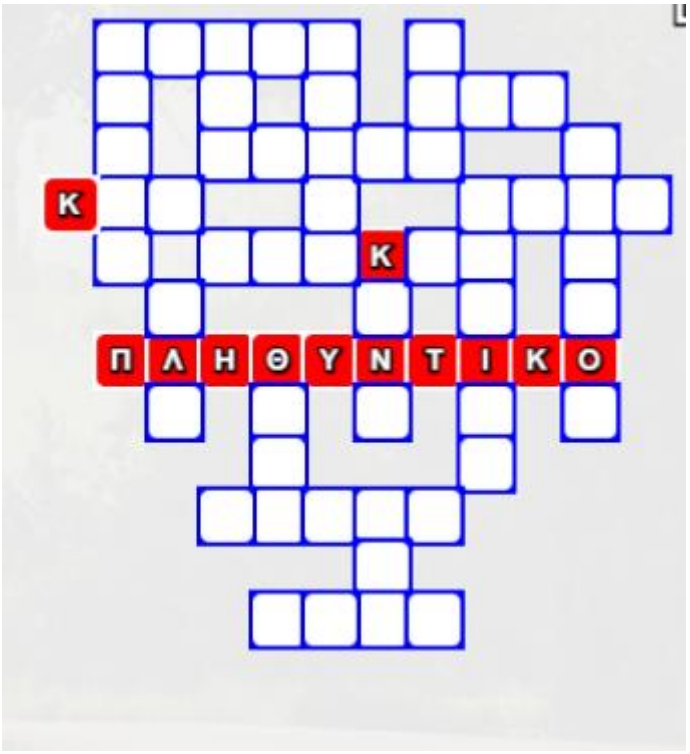
Η λειτουργία βοήθειας με το εικονίδιο της λάμπας επιτρέπει στον χρήστη να ξεκλειδώσει ολόκληρη μία λέξη με μία μόνο κίνηση — ιδανική όταν έχει κολλήσει σε ένα ολόκληρο τμήμα του πλέγματος.Ο σκοπός της είναι να εμφανιστούν αμέσως όλα τα γράμματα μίας μη-βρεθείσας λέξης, χωρίς να χρειάζεται να αποκαλύψεις γράμμα-γράμμα. Η ενεργοποίηση της γίνεται απλά πατώντας το εικονίδιο της λάμπας δίπλα στο Hint.

Αμέσως εμφανίζεται παράθυρο επιβεβαίωσης: “Θέλετε να αποκαλύψετε την επόμενη λέξη; Κόστος 10 πόντοι.” Και δίνεται η επιλογή να πατήσει ο χρήστης ΝΑΙ ή ΟΧΙ.



Εικόνα 31: Παράθυρο βοήθειας λάμπας

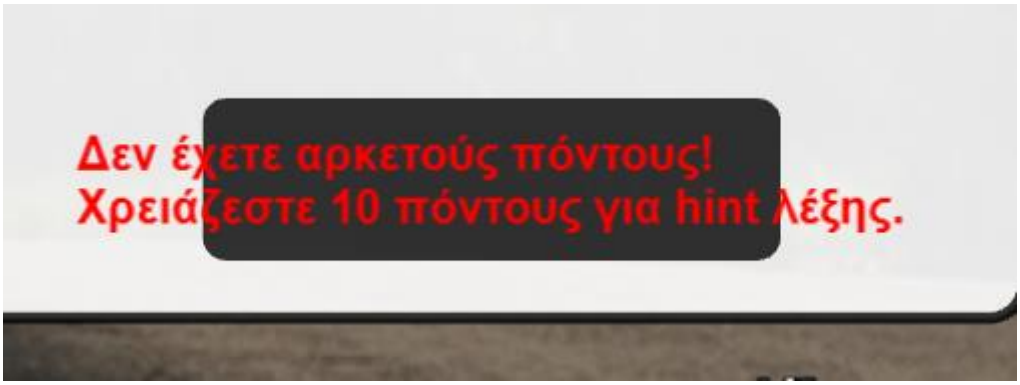
Μόλις ο χρήστης πατήσει ΝΑΙ το σκορ του μειώνεται κατά -10 πόντους.



Εικόνα 32: Πριν την βοήθεια

Εικόνα 33: Συμπλήρωση ΗΘΙΚΟΥ με Βοήθεια

Η λέξη καταχωρείται στη δεξιά λίστα ως “βρεθείσα” αλλά **δεν λαμβάνεις πόντους** γι’ αυτή—θεωρείται βοήθεια. Είναι χρήσιμη βοήθεια όταν σσαν χρήστης ε χωρίζουν πολλά γράμματα από τη λύση και η αποκάλυψη γράμματος-γράμματος είναι χρονοβόρα, επίσης σε λέξεις με ασυνήθιστους συνδυασμούς ή πολύ μεγάλα μήκη, όπου ένα Hint ίσως δεν αρκεί. Αν ο χρήστης δεν έχει πόντους για να χρησιμοποιήσει την βοήθεια τότε εμφανίζεται παράθυρο που να τον ενημερώνει ότι δεν έχει τους πόντους για την χρήση της.



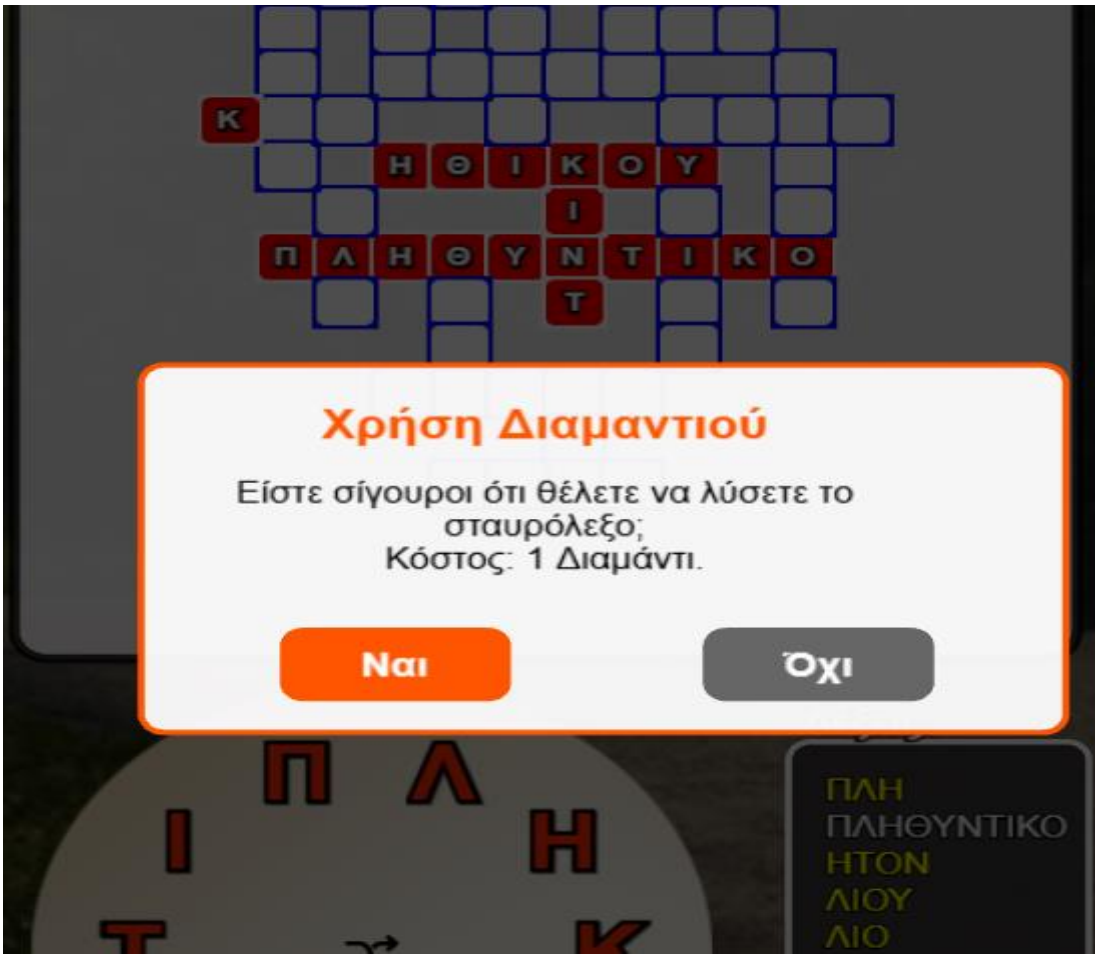
Εικόνα 34: Pop-up ενημέρωσης

3.8 Άμεση Λύση Επιπέδου

Η λειτουργία διαμαντιού επιτρέπει να ολοκληρώσει ο χρήστης το σταυρόλεξο σε ένα βήμα, χρησιμοποιώντας ένα νόμισμα του παιχνιδιού.Σκοπός είναι να ξεκλειδωθούν όλα τα γράμματα και να καταχωρηθούν όλες οι λέξεις του επιπέδου αμέσως, χωρίς καμία περαιτέρω επιλογή. Η ενεργοποίηση γίνεται απλά πατώντας το διαμάντι τέρμα πανώ αριστερά.

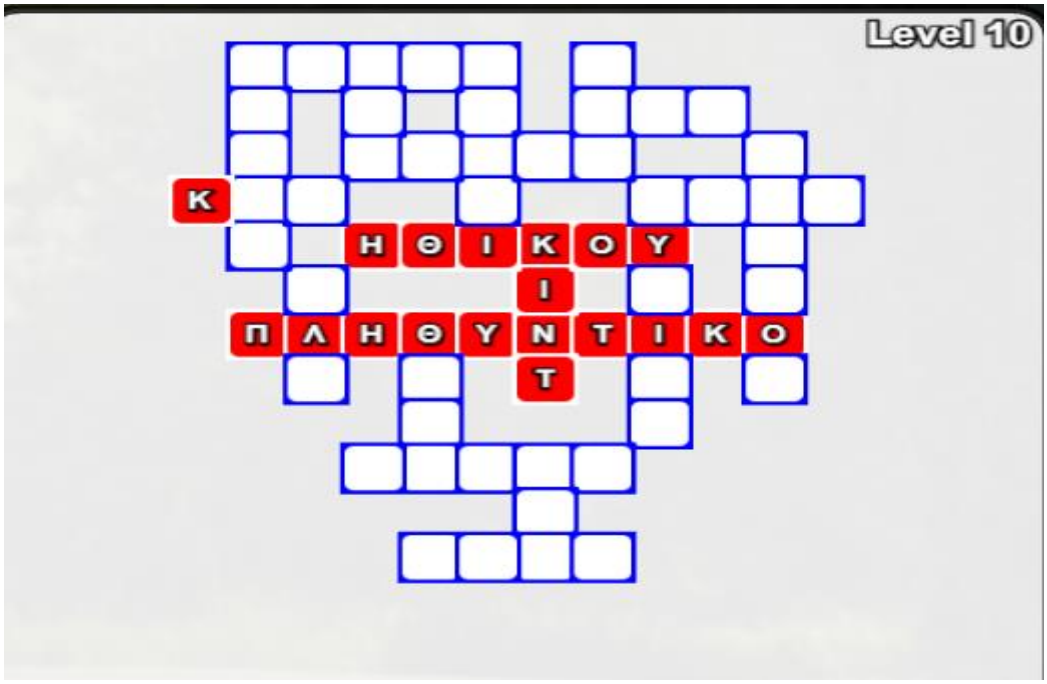
Εικόνα 35: Βοήθεια Διαμαντιού

Μόλις ο χρήστης πατήσει το διαμάντι εμφανίζεται παράθυρο επιβεβαίωσης. Για να αποφασίσει ο χρήστης αν θέλει τελικά την χρήση της βοήθειας.



Εικόνα 36: Παράθυρο επιβεβαίωσης

Αν ο χρήστης αποφασίσει ότι θέλει να χρησιμοποιήσει την βοήθεια τότε αφαιρείται από τα διαμάντια που έχει ήδη -1 διαμάντι. Έτσι λύνεται όλο το επίπεδο.



Εικόνα 37: Πριν την χρήση διαμαντιού

Εικόνα 38: Μετά την χρήση διαμαντιού

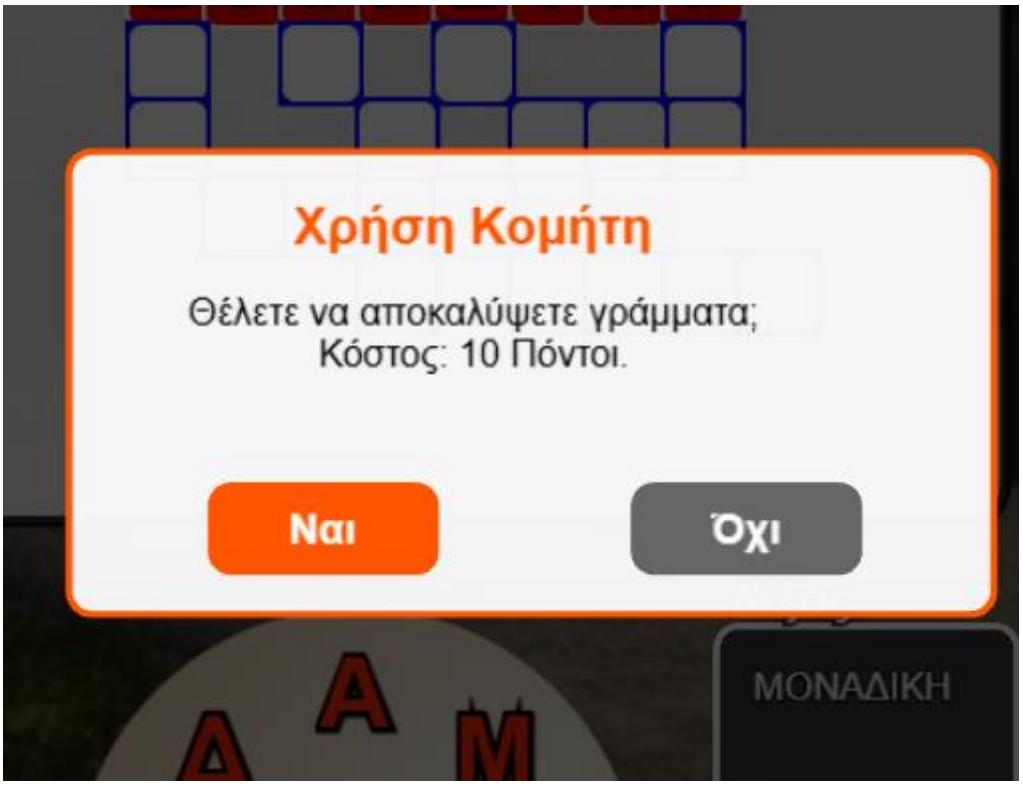
Το επίπεδο σηματοδοτείται ως ολοκληρωμένο και, μετά από μικρή καθυστέρηση, ξεκινάει το επόμενο. Όλες οι κύριες λέξεις (από το πλέγμα) θεωρούνται "βρεθείσες" και προστίθενται με τη σειρά τους στη δεξιά λίστα. Και όλα τα κελιά έχουν συμπληρωθεί με τις κατάλληλες λέξεις. Με την χρήση του διαμαντιού έχει ο χρήστης την απόλυτη «ασφάλεια» να τελειώνει το επίπεδο αμέσως, μεταφέροντας το σκορ και τα διαμάντια του στο επόμενο χωρίς κόπο.

3.9 Αποκάλυψη Γραμμάτων

Η λειτουργία των κομήτων αποτελεί ένα συλλογικό hint: αποκαλύπτει ένα γράμμα σε κάθε λέξη ταυτόχρονα — ιδανικό όταν το επίπεδο έχει πολλές λέξεις και θες γρήγορη πρόοδο.Σκοπός είναι να δώσει στον παίκτη μια "εκτόξευση" στην επίλυση, αποκαλύπτοντας ένα γράμμα από **κάθε** μη-βρεθείσα λέξη με ένα μόνο πάτημα. Η συγκεκριμένη βοήθεια εμφανίζεται **μόνο** σε επίπεδα με πάνω από 7 λέξεις.

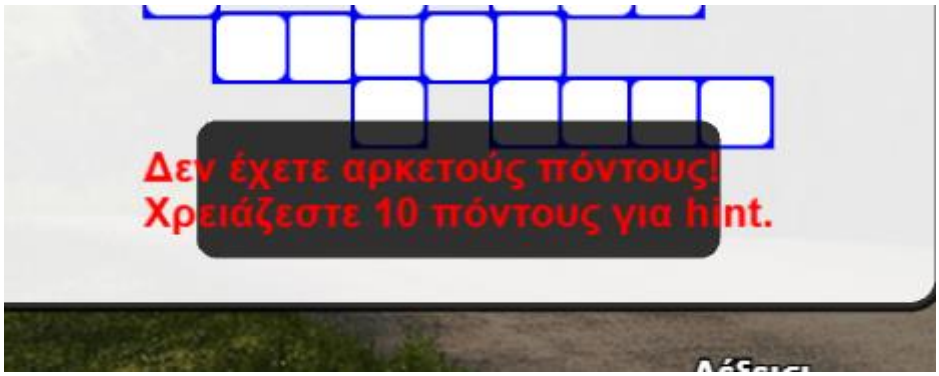
Εικόνα 39: Βοήθεια Κομήτη

Πατώντας το, ανοίγει παράθυρο επιβεβαίωσης: “Θέλετε να αποκαλύψετε γράμματα σε κάθε λέξη; Κόστος: 10 πόντοι.”



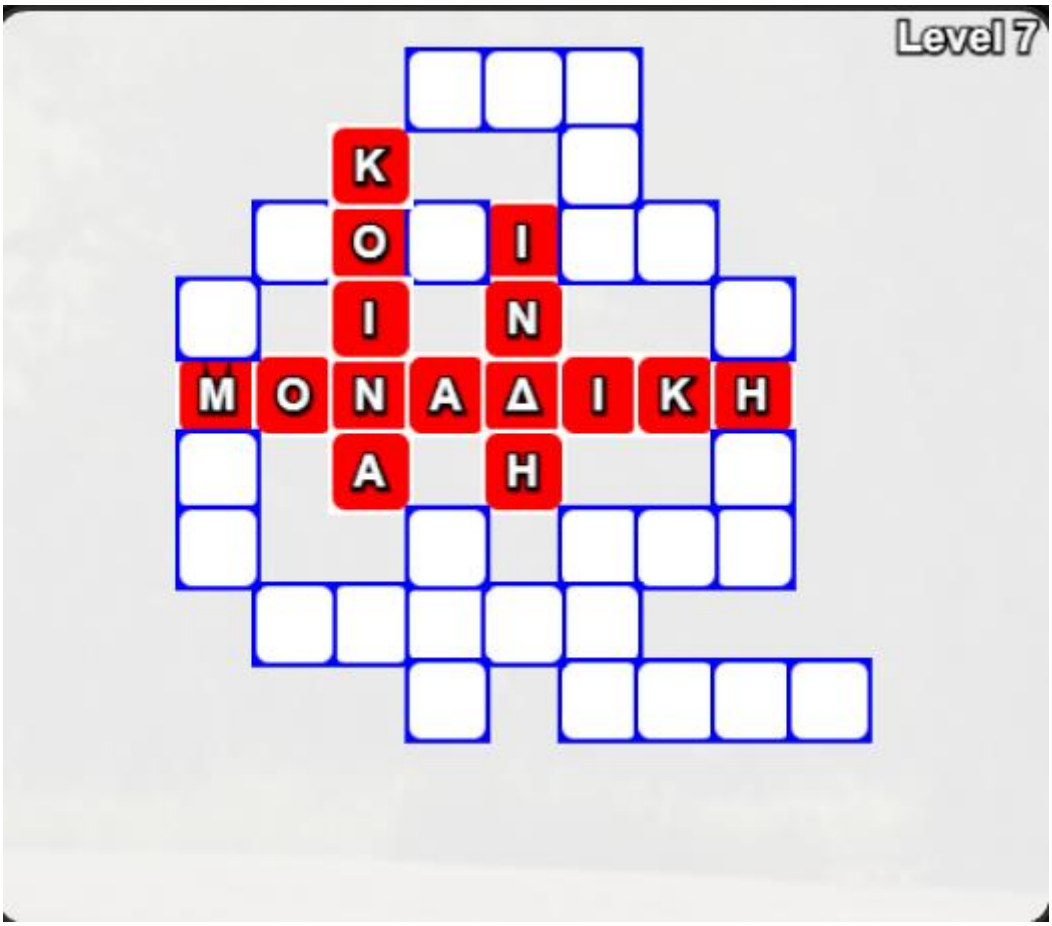
Εικόνα 40: Παράθυρο επιβεβαίωσης

Αν δεν έχει τουλάχιστον 10 πόντους, εμφανίζεται μήνυμα “Δεν έχετε αρκετούς πόντους!” και η ενέργεια ακυρώνεται.

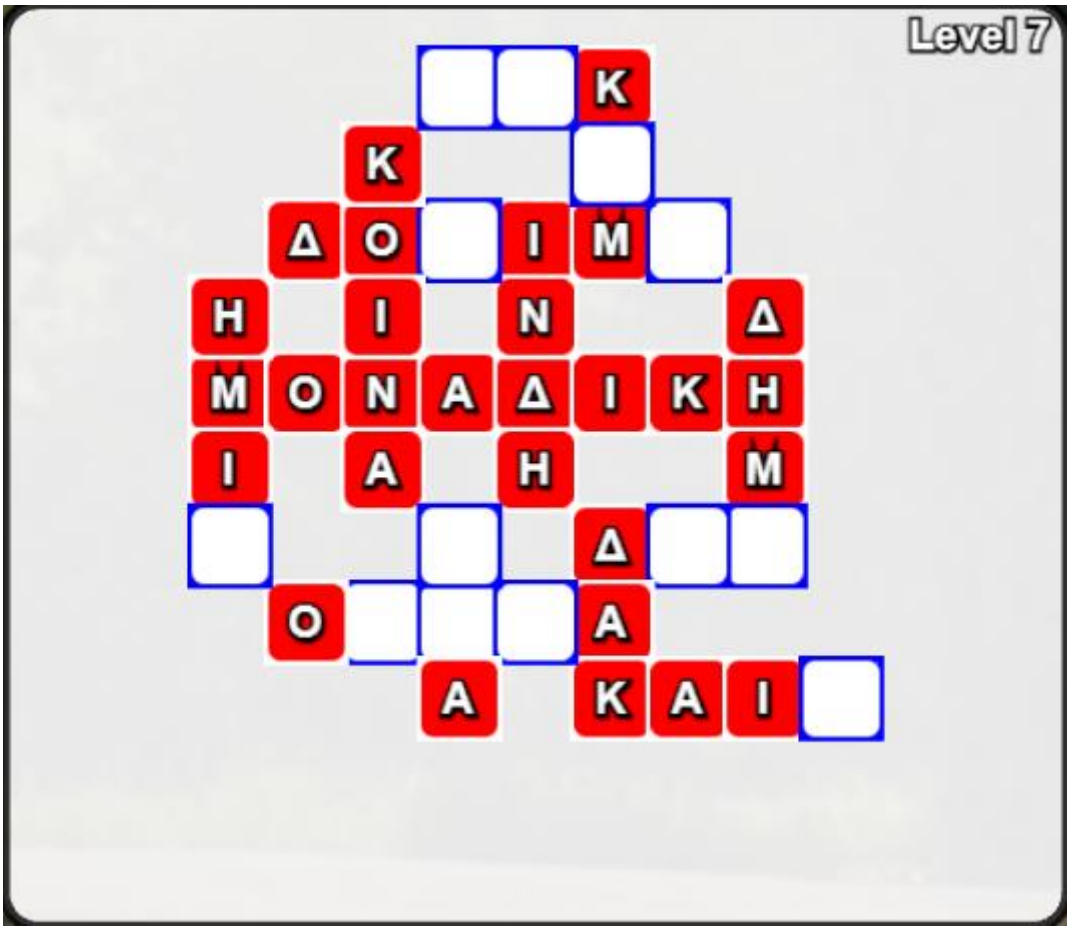


Εικόνα 41: Ενημέρωση χρήστη

Κάθε χρήση αφαιρεί **10 πόντους** από το συνολικό σου σκορ. Για **κάθε** λέξη που δεν έχει βρεθεί, αποκαλύπτεται ένα τυχαίο κλειδωμένο γράμμα. Με αυτόν τον τρόπο, όλες οι λέξεις προχωρούν ταυτόχρονα, χωρίς να επιλέγεις ξεχωριστά.



Εικόνα 42: Πριν την χρήση βοήθειας



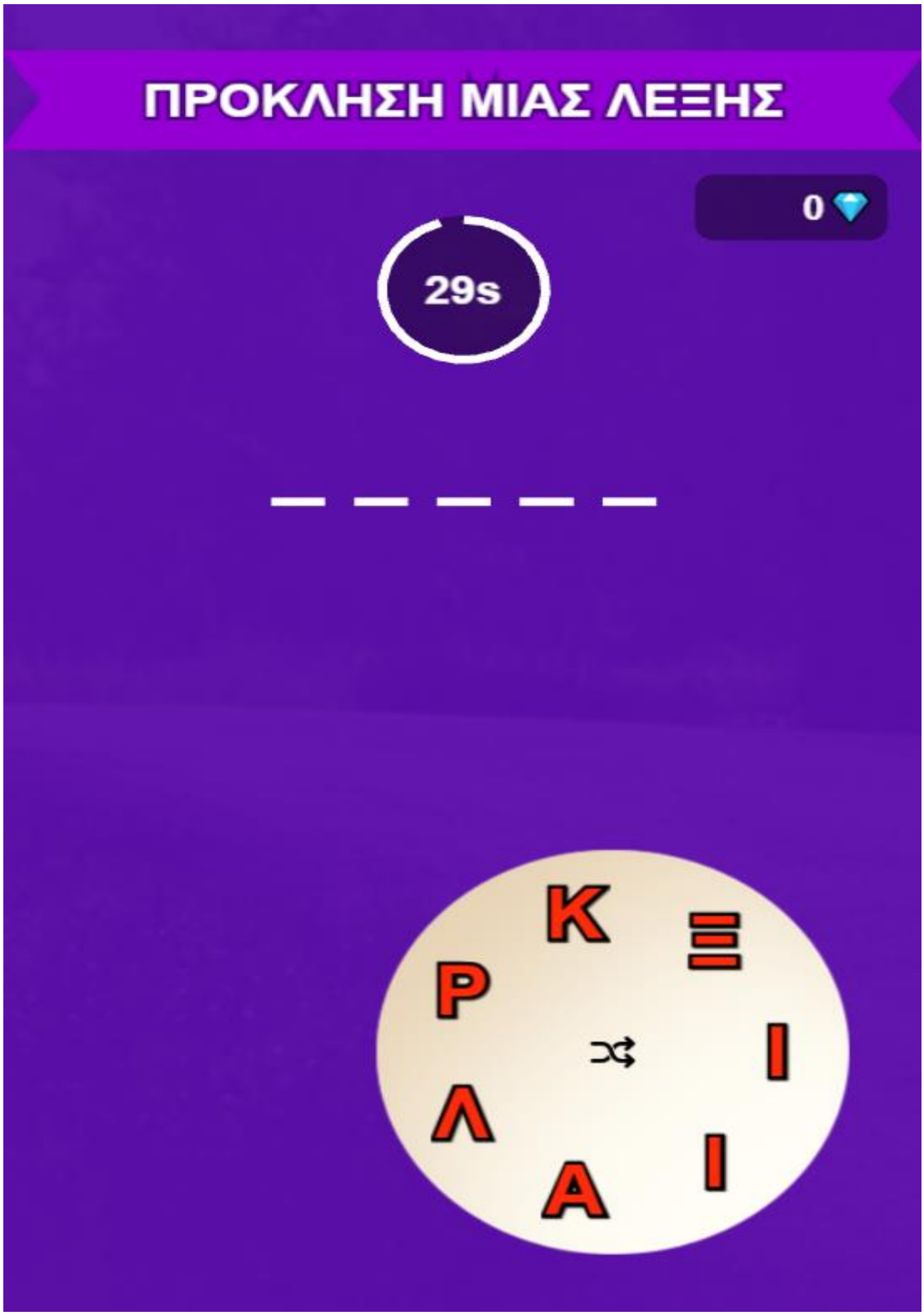
Εικόνα 43: Χρήση Κομήτων

Με τη χρήση της συγκεκριμένης βοήθειας ο χρήστης ενισχύει δραμαματικά την αποκάλυψη των λέξεων όταν το παιχνίδι γίνεται πιο πολύπλοκο, διατηρώντας όμως τον έλεγχο του ρυθμού και των πόντων σου.

3.10 Bonus Level

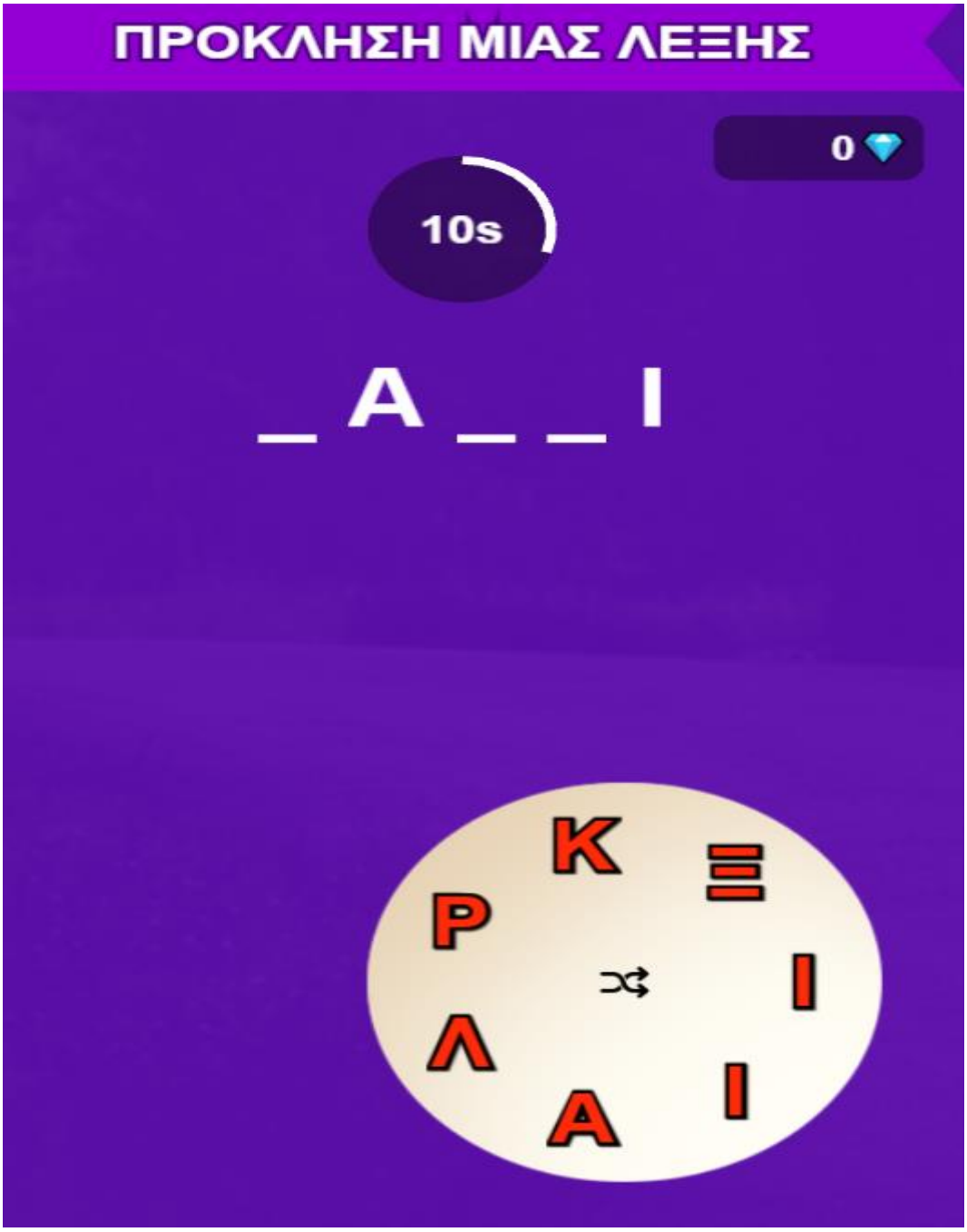
Τα επίπεδα bonus είναι ειδικές προκλήσεις που ξεκλειδώνουν μετά από συγκεκριμένους στόχους, συγκεκριμένα κάθε τρία επίπεδα υπάρχει ενα επιπεδο έξτρα.Το σκεπτικό του επιπέδου αυτού είναι να δώσει διαμάντια στον χρήστη αφού βρει μία λέξη συνήθως 5–7 γράμματα, σχεδιασμένα για στοχευμένο σχηματισμό λέξης. Έτσι και η παλέτα διαθέτει τον κατάλληλο αριθμό γραμμάτων.

Το περιβάλλον του έξτρα επιπέδου είναι μία μπλε-μωβ σκηνή, η οποία έχει τον αριθμό των διαμαντιών που έχει ήδη ο χρήστης μία παλέτα γραμμάτων για την λέξη, παύλες για τα γράμματα της λέξης που πρέπει να βρει ο χρήστης και ένα χρονόμετρο.



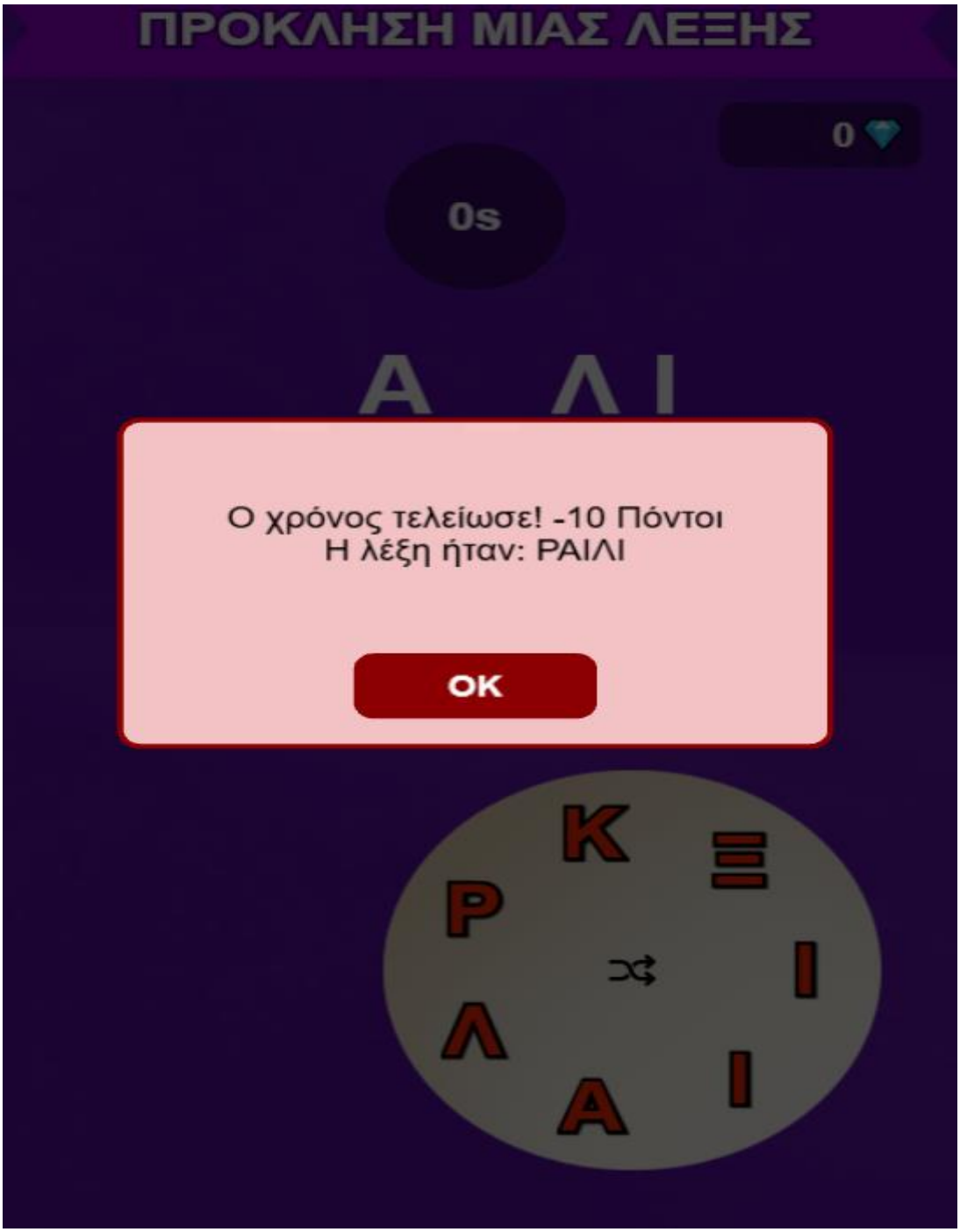
Εικόνα 44: Περιβάλλον έξτρα επιπέδου

Το χρονόμετρο έχει χρόνο 30 δευτερολέπτων και μετράει αντίστροφα μέχρι να βρει την λέξη ο χρήστης. Όσο ο χρόνος μειώνεται αρχίζουν και βγαίνουν γράμματα ώστε να βοηθήσουν τον χρήστη.



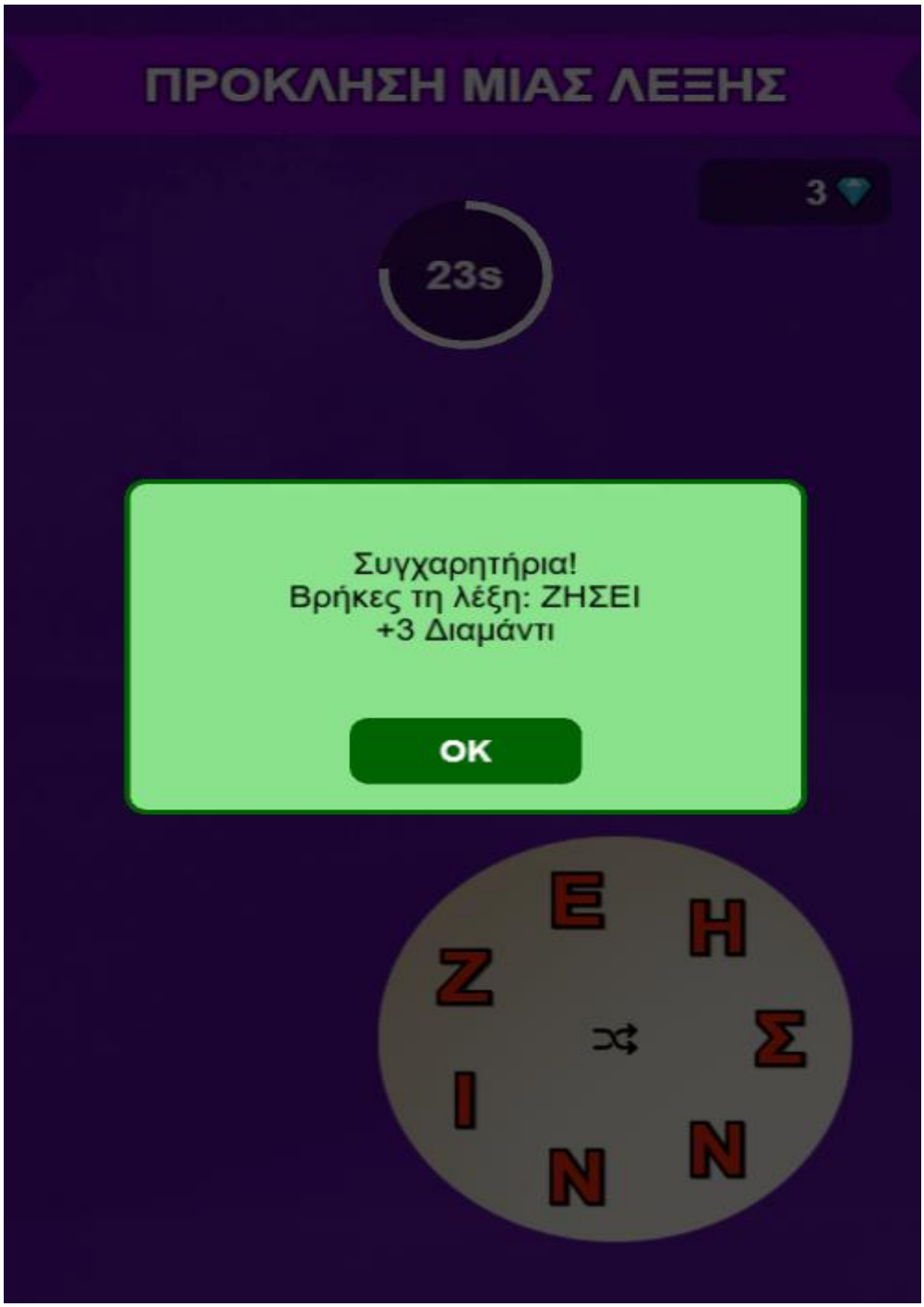
Εικόνα 45: Εμφάνιση γραμμάτων

Αν ο χρόνος τελειώσει και ο χρήστης δεν έχει βρει την λέξη τότε χάνει πόντους. Αν έχει ήδη διαμάντια τότε χάνει -1 διαμάντι, αλλιώς χάνει -10 πόντους από το βασικό σκορ.Ο χρήστης ενημέρωνεται με την απώλεια των βαθμών και με το ποιά ήταν η λέξη που δεν βρήκε μέσω ενός παραθύρου που εμφανίζεται αφού τελειώσει ο χρόνος.



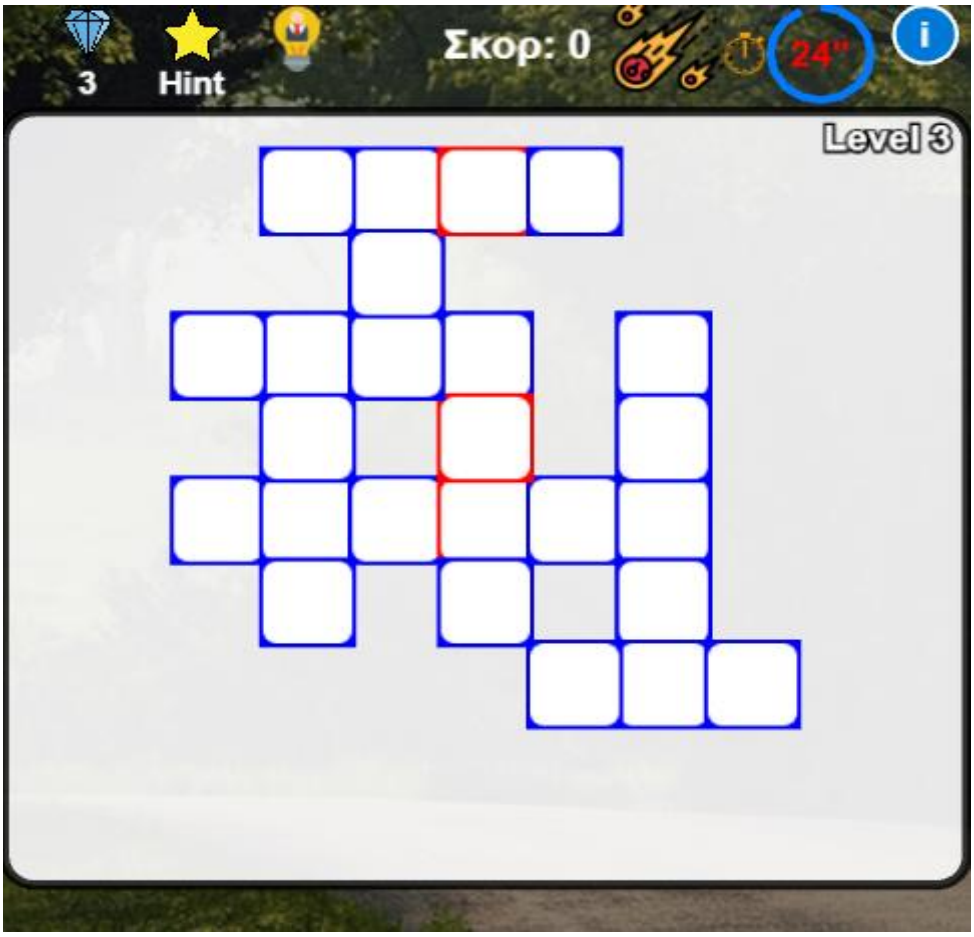
Εικόνα 46: Αφαίρεση πόντων και εμφάνιση λέξης

Με τον ίδιο τρόπο ενημέρωνεται ο χρήστης όταν βρει την λέξη που ψάχνει το έξτρα επίπεδο.



Εικόνα 47: Προσθήκη διαμαντιών

Πατώντας το ΟΚ πάει στο επόμενο επίπεδο και εκεί βλέπουμε ότι έχει αλλάξει ο αριθμός των διαμαντιών.



Εικόνα 48: Διαμάντια από Bonus Level

Κεφάλαιο 4ο: Ανάλυση Τεχνολογιών

Η ανάπτυξη του παιχνιδιού βασίστηκε εξ ολοκλήρου σε τεχνολογίες ιστού, αξιοποιώντας κυρίως τη γλώσσα προγραμματισμού JavaScript και εξειδικευμένες βιβλιοθήκες για τη γραφική αναπαράσταση και τη διαχείριση κινούμενων αντικειμένων. Η επιλογή των τεχνολογιών έγινε με γνώμονα τη φορητότητα, την απόδοση, την επεκτασιμότητα και την προσβασιμότητα της εφαρμογής.

4.1 JavaScript

Η επιλογή του σύγχρονου JavaScript (ES6+) με modules, async/await, ιδιωτικά πεδία (private class fields) και arrow functions μας επιτρέπει να γράφουμε κώδικα σαφή και αρθρωτό. Κάθε σκηνή (π.χ. LevelScene, BonusScene) είναι ανεξάρτητη κλάση, που υλοποιεί τη δική της λογική, το state και τα callbacks. Η χρήση async methods για το φόρτωμα λεξικού εξασφαλίζει ότι δεν μπλοκάρει ο κεντρικός thread του παιχνιδιού, ενώ τα modules διαχωρίζουν ευδιάκριτα τις ευθύνες: rendering, game logic, generation επιπέδων, loading assets.

4.2 Pixi.js

Η PixiJS είναι μια ισχυρή JavaScript βιβλιοθήκη για rendering γραφικών σε HTML5 Canvas και WebGL, που έχει σχεδιαστεί ειδικά για gaming και διαδραστικές εφαρμογές. Στο έργο μας, χρησιμοποιούμε PixiJS για να οργανώσουμε κάθε οπτικό στοιχείο σε «Containers» (ομαδοποιημένα layers)· έτσι το πλέγμα του σταυρόλεξου, τα πλαίσια, τα κείμενα και τα εικονίδια τοποθετούνται, μετακινούνται και ζωγραφίζονται αποδοτικά. Η δυνατότητα να σχεδιάζουμε δυναμικά γεωμετρικά σχήματα με τα Graphics αντικείμενα μάς επιτρέπει να δημιουργούμε φόντα, περιγράμματα και καμπύλες (όπως ο δακτύλιος του χρονομέτρου) με πλήρη έλεγχο χρώματος, διαφάνειας και stroke. Παράλληλα, με την ενσωματωμένη διαχείριση βάθους (zIndex) εξασφαλίζουμε ότι κάθε στοιχείο εμφανίζεται πάνω ή κάτω από τα υπόλοιπα με την κατάλληλη σειρά, χωρίς πολύπλοκο manual layering.

Η PixiJS διαθέτει έναν built-in loader που διαχειρίζεται σειριακά ή παράλληλα το φόρτωμα εικόνων, JSON, spritesheets και άλλων πόρων. Στη φάση initialization, πριν εμφανιστεί το πρώτο επίπεδο, ο loader ανακτά όλα τα αρχεία που χρειάζονται: sprites κουμπιών, εικόνες διαμαντιών, το JSON του λεξικού, ακόμα και custom spritesheets (π.χ. για κουμπιά hover). Μέχρι να ολοκληρωθεί το load, εμφανίζουμε ένα απλό splash screen ή loading bar, εξασφαλίζοντας ότι κατά το παιχνίδι δεν θα εμφανιστούν «κενά» frames ή flickering.

4.3 GSAP (GreenSock Animation Platform)

Η GSAP χρησιμοποιήθηκε για τη δημιουργία animations στα γραφικά αντικείμενα του παιχνιδιού. Εφαρμόζεται σε καταστάσεις όπως:

- Μετάβαση κουμπιών
- Εμφάνιση/εξαφάνιση σκηνών
- Εφέ ολοκλήρωσης επιπέδου

Η συνδυαστική χρήση Pixi.js και GSAP επιτρέπει fluid εμπειρία χρήστη με οπτική συνέπεια και καλή απόκριση.

Η GSAP είναι de-facto το industry standard για animation στο web: δεν περιορίζεται σε CSS transitions αλλά προσφέρει ελεγχόμενα tweens σε οποιαδήποτε JavaScript ιδιότητα. Στο παιχνίδι, αξιοποιούμε την GSAP για να «παγώνουμε» αριθμητικές μεταβλητές στον χρόνο – όπως το countdown των 25 δευτερολέπτων για τα ειδικά κουτιά – να ενημερώνουμε αδιάλειπτα το κείμενο και να σχεδιάζουμε τον κινούμενο δακτύλιο απέναντι στο background ring.

Με callbacks onComplete χειριζόμαστε την οριστική λήξη του χρονομέτρου, καταργώντας τα γραφικά και απενεργοποιώντας την ανταμοιβή διαμαντιών, ενώ η ευκολία χρήσης του ease:"none" μάς δίνει απόλυτη γραμμικότητα της κίνησης.

4.4 Webpack & Babel

Για να διανεμηθεί εύκολα η εφαρμογή, το Webpack «πακετάρει» όλα τα JavaScript modules, τα JSON λεξικά, τα sprites και το CSS σε ένα ή δύο τελικά bundles, βελτιστοποιημένα για παραγωγή (minification, tree-shaking). Ο Babel μεταγλωττίζει τον ES6+ κώδικα σε συμβατό JavaScript που τρέχει απρόσκοπτα ακόμα και σε παλαιότερους browsers, χωρίς να χρειάζεται ο χρήστης να αναβαθμίσει το περιβάλλον του.

Επιπλέον, χάρη στο hot-module reloading σε φάση ανάπτυξης, μπορούμε να δοκιμάζουμε αλλαγές στον κώδικα και να βλέπουμε το αποτέλεσμα σε πραγματικό χρόνο.

4.5 Δημιουργία Λεξικού με Python

Για τη βελτίωση της ποιότητας του λεξικού λέξεων που χρησιμοποιεί το παιχνίδι, αναπτύχθηκε ένα βοηθητικό εργαλείο με χρήση της γλώσσας Python. Το εργαλείο αυτό, μέσω του αρχείου get_words.py, δημιουργεί αυτόματα ένα νέο σύνολο καθαρών, ελληνικών λέξεων, κατάλληλων για χρήση στο gameplay.

Η λειτουργία του script περιλαμβάνει, λήψη των πιο συχνών ελληνικών λέξεων, κανονικοποίηση και καθαρισμός, φιλτράρισμα , εξαγωγή σε αρχείο JSON, σημασία στο gameplay.

Η λήψη των πιο συχνών ελληνικών λέξεων γίνεται με την χρήση της βιβλιοθήκης wordfreq.

Αυτό εξασφαλίζει ότι οι λέξεις που θα περιληφθούν στο λεξικό είναι πραγματικά κοινές και χρήσιμες στον παίκτη.Η κανονικοποίηση και ο καθαρισμός εξασφαλίζουν την εκγυρότητα και την καθαρότητα των λέξεων, εφαρμόζεται κανονικοποίηση Unicode, η οποία αφαιρεί τόνους και μετατρέπει όλους τους χαρακτήρες σε κεφαλαία.

Η διαδικασία μετατρέπει, για παράδειγμα, τη λέξη «αγάπη» σε «ΑΓΑΠΗ».Έπειτα ακολουθεί το φιλτράρισμα ώστε να διατηρηθούν μόνο οι λέξεις που, έχουν τουλάχιστον 3 γράμματα και περιέχουν μόνο ελληνικά γράμματα (χωρίς αριθμούς ή σύμβολα).

Τέλος, οι καθαρές λέξεις αποθηκεύονται σε μορφή JSON, με δομή συμβατή με το dictionary.json του παιχνιδιού.Το παραγόμενο αρχείο μπορεί να μετονομαστεί σε dictionary.json και να αντικαταστήσει το παλιό λεξικό στην εφαρμογή.

Η χρήση καθαρού λεξικού ενισχύει την ποιότητα του gameplay:

- Αποτρέπει λέξεις με αριθμούς, σύμβολα ή σπάνια ορθογραφία
- Εξασφαλίζει ότι οι παίκτες βρίσκουν μόνο κοινές και αναγνωρίσιμες λέξεις
- Βελτιώνει την αλγοριθμική ανάλυση κατά τη δημιουργία επιπέδων

4.6 HTML και CSS (υποστηρικτικά)

Αν και το κύριο UI σχεδιάζεται μέσω Pixi.js, γίνεται χρήση απλού HTML/CSS για την τοποθέτηση του καμβά, τη φόρτωση πόρων και βασικά στοιχεία wrapper στο περιβάλλον του

browser.Η συνδυαστική χρήση των παραπάνω τεχνολογιών καθιστά την εφαρμογή αποδοτική, εύχρηστη και πλήρως επεκτάσιμη για μελλοντικές αναβαθμίσεις, ενώ διατηρεί τον κώδικα καθαρό και κατανοητό.Η αρχιτεκτονική της εφαρμογής είναι βασισμένη σε αρθρωτή (modular) προσέγγιση, όπου κάθε ενότητα του παιχνιδιού υλοποιείται σε ξεχωριστό αρχείο ή module, διασφαλίζοντας έτσι την επεκτασιμότητα, την ευκολία συντήρησης και τη σαφήνεια στη ροή των λειτουργιών.

Ο κώδικας είναι διαχωρισμένος σε βασικά αρχεία με διακριτές αρμοδιότητες, όπως index.js, game.js, LevelGenerator.js, wordGenerator.js, BonusScene.js, LevelScene.js, gridSection.js, letterPalete.js κ.ά. Κάθε ένα από αυτά διαχειρίζεται μία συγκεκριμένη λειτουργία του παιχνιδιού, από τη φόρτωση των δεδομένων έως την εμφάνιση της διεπαφής και την κατασκευή των επιπέδων.

4.7 Το αρχείο index.js – Σημείο εκκίνησης

Το index.js αποτελεί το σημείο εισόδου της εφαρμογής. Εκεί αρχικοποιείται η μηχανή του Pixi.js, φορτώνονται τα assets (π.χ. εικόνες, ήχοι), και ορίζονται τα βασικά στοιχεία του καμβά. Επίσης, ενεργοποιείται η πρώτη σκηνή του παιχνιδιού.

4.8 LocalStorage API

Χρησιμοποιούμε τον browser’s LocalStorage για να αποθηκεύουμε το ιστορικό του χρήστη—τα sets γραμμάτων που έχει ήδη ολοκληρώσει—και να τα ανακτούμε σε επόμενη επίσκεψη. Με JSON stringify/parse διατηρούμε ένα array canonical strings, που ελέγχεται πριν από κάθε νέα γενιά επιπέδου ώστε να μην υπάρχουν διπλοεγγραφές. Αυτό δίνει συνεχή πρόοδο χωρίς server και επιτρέπει offline λειτουργία.

4.9 DevTools Integration

Για debugging και live inspection έχουμε προσθέσει στον constructor ένα global reference (window.currentLevelScene = this). Έτσι, σε οποιαδήποτε στιγμή ανοίγουμε τα Chrome DevTools και γράφουμε currentLevelScene.gridSection.dictionary ή currentLevelScene.specialCountByWord, μπορούμε να δούμε το τρέχον state: ποιες λέξεις είναι ξεκλείδωτες, ποια extras έχουν βρεθεί, ποιοι timers τρέχουν και πόσα διαμάντια έχει ο παίκτης. Αυτό είναι ανεκτίμητο για επίδειξη σε παρουσίαση ή για ταχύτερη επίλυση προβλημάτων κατά την ανάπτυξη.

Κεφάλαιο 5ο: Αρχιτεκτονική της εφαρμογής

5.1 Το αρχείο LevelScene.js – Κεντρική σκηνή παιχνιδιού

Το LevelScene.js περιγράφει την κεντρική σκηνή, όπου ο χρήστης αλληλεπιδρά με το grid και την παλέτα γραμμάτων. Διαχειρίζεται το gameplay, την καταγραφή των λέξεων, το animation, και τον έλεγχο της προόδου. Το LevelScene.js αποτελεί την καρδιά της λειτουργικότητας του παιχνιδιού, καθώς είναι υπεύθυνο για τη δημιουργία, την εμφάνιση και την αλληλεπίδραση του χρήστη με το εκάστοτε επίπεδο. Εδώ φορτώνονται τα δεδομένα του επιπέδου (λέξεις, παλέτα γραμμάτων, extra λέξεις) και ξεκινά η διεπαφή του παίκτη με το grid και την παλέτα.

Λειτουργίες της σκηνής:

- Δημιουργία δυναμικού grid σταυρολέξου μέσω του GridSection, με βάση τις λέξεις του επιπέδου.
- Τοποθέτηση της παλέτας γραμμάτων με προσαρμοζόμενη ακτίνα και κεντράρισμα στην οθόνη.
- Ενημέρωση του σκορ και των πόντων σε πραγματικό χρόνο.
- Εμφάνιση των κουμπιών βοήθειας (hint γράμματος, hint λέξης, solve).
- Παρακολούθηση των λέξεων που βρέθηκαν ή απομένουν, ώστε να ελέγχεται η ολοκλήρωση του επιπέδου.

Επιπλέον δυνατότητες:

- Ενσωμάτωση animation GSAP για smooth μετάβαση από/προς τη σκηνή.
- Αρχικοποίηση counters για σκορ και διαμάντια.
- Καταγραφή χρήσης βοήθειας ώστε να επηρεάζεται το σκορ.

5.2 Το αρχείο gridSection.js – Γραφικά στοιχεία

Στο gridSection.js διαχειρίζεται το πλέγμα του σταυρόλεξου. Δημιουργεί τα κουτιά και τα γεμίζει με γράμματα όταν εντοπιστεί σωστή λέξη. Με λίγα λόγια είναι το module που “χτίζει” και ελέγχει το ίδιο το πλέγμα του σταυρόλεξου:

Κατασκευή Πλέγματος: Σαρώνει τα δεδομένα κάθε λέξης (θέσεις, κατευθύνσεις), υπολογίζει δυναμικά μέγεθος και κεντράρισμα, και φτιάχνει τα grid cells με το σωστό γράμμα.

Παλέτα Γραμμάτων: Συνδέει την επάνω μπάρα επιλογής γραμμάτων (LetterPalette) με callback ώστε, όταν ο χρήστης σχηματίζει λέξη, να ελέγχεται αν είναι έγκυρη σε σταυρόλεξο, extra ή λεξικό.

Λεξικό & Extra: Φορτώνει ασύγχρονα το dictionary.json και κρατά λίστα “extra” λέξεων που δίνουν επιπλέον πόντους.

Βασική Λογική: Ελέγχει αν μια υποβληθείσα λέξη έχει ήδη βρεθεί, αν είναι εντός πλέγματος, αν είναι extra ή αν απλώς ανήκει στο λεξικό – και καλεί τα αντίστοιχα callbacks.

5.3 Το αρχείο BonusScene.js – Bonus γύρος

Το σκηνικό της πρόκλησης «μία λέξη» με περιορισμένο χρόνο:

UI Διαφορετικού Τύπου: Με ξεχωριστή background χρωματική παλέτα, banner τίτλου, δακτύλιο αντίστροφης μέτρησης και μεγάλα underscore για το στόχο.

Ροή Χρόνου & Υποδείξεις: Ο αντίστροφος χρονοδιακόπτης αποκρύπτει δακτύλιο, ενημερώνει το υπόλοιπο δευτ. και κάθε 10 δευτ. «αποκαλύπτει» τυχαίο γράμμα, μειώνοντας ταυτόχρονα τη μέγιστη ανταμοιβή διαμαντιών.

Ολοκλήρωση & Ποινές: Αν βρεθεί σωστά η λέξη εγκαίρως, δίνει διαμάντια, αλλιώς αφαιρεί διαμάντια ή πόντους όταν λήξει ο χρόνος ή αποκαλυφθούν όλα τα γράμματα από hints.

5.4 Το αρχείο LevelGenerator.js

Το backbone της παραγωγής περιεχομένου, δημιουργεί αυτόματα επίπεδα με αυξανόμενη δυσκολία:

Γεννήτρια Pangram: Επιλέγει λέξεις-«pangram» που περιέχουν μοναδικά όλα τα γράμματα της palette.

Παλέτα & Ιστορικό: Διαμορφώνει παλέτες γραμμάτων βάσει συχνότητας φωνηέντων/σύμφωνων, και αποθηκεύει σε localStorage τα σύνολα γραμμάτων που έχουν ήδη χρησιμοποιηθεί, για να μην επαναλαμβάνεται κάποιο επίπεδο.

Δημιουργία Λίστας Λέξεων: Με τη βοήθεια της βιβλιοθήκης wordGenerator, βρίσκει όλες τις δυνατές λέξεις μεταξύ 3–max γραμμάτων και φτιάχνει τον κατάλογο «other» λέξεων.

Crossword Layout: Στέλνει στο generateCrossword τα ζητούμενα σύνολα λέξεων και το πλέγμα μεγέθους ώστε να τοποθετηθούν οι λέξεις με το σωστό format για το παιχνίδι.

5.5 Διαχείριση Δεδομένων και Λεξικού

Το λεξικό της εφαρμογής φορτώνεται από το αρχείο dictionary.json, το οποίο παράγεται με χρήση Python script (get_words.py). Αυτό το script:

- Λαμβάνει τις πιο συχνές ελληνικές λέξεις.
- Αφαιρεί τόνους, πεζά γράμματα, σύμβολα.
- Εξάγει το λεξικό σε κατάλληλη μορφή JSON.

Η σωστή οργάνωση και διαχείριση των δεδομένων είναι καθοριστική για την αξιοπιστία και την απόδοση της εφαρμογής. Στο επίκεντρο αυτής της διαδικασίας βρίσκεται το ελληνικό λεξικό, που αποτελείται από δεκάδες χιλιάδες λέξεις χωρίς τόνους και με μήκος 3–12 γραμμάτων. Μετά την αρχική δημιουργία του JSON αρχείου (μέσω του Python script και της βιβλιοθήκης wordfreq), το dictionary.json φορτώνεται ασύγχρονα στην έναρξη κάθε επιπέδου, αποφεύγοντας το "πάγωμα" της διεπαφής. Κατά το φόρτωμα, οι λέξεις φιλτράρονται για να διασφαλιστεί ότι πληρούν τα κριτήρια μορφολογίας (μόνο ελληνικά γράμματα, καμία αριθμητική ή ειδική μορφή, κατάλληλο μήκος). Στη συνέχεια, το λεξικό αποθηκεύεται σε μία μεταβλητή στη μνήμη του browser, ώστε κάθε έλεγχος εγκυρότητας — είτε πρόκειται για τη βασική λίστα λέξεων του σταυρόλεξου, είτε για τα «extra» ή τις λεξικογραφικές υποβολές του παίκτη — να γίνεται ταχύτατα.

Παράλληλα, η ιστορία των επιπέδων (τα σύνολα γραμμάτων που έχουν ήδη χρησιμοποιηθεί) φυλάσσεται στο LocalStorage. Με αυτόν τον τρόπο, όταν ο παίκτης εγκαταλείψει και ξαναμπεί στην εφαρμογή, το σύστημα δεν θα του ξαναδώσει επίπεδα που έχει ήδη λύσει, διατηρώντας την πρόοδό του με τη μορφή canonical keys. Η ανάκτηση και ενημέρωση γίνεται με απλό JSON parse/stringify, εξασφαλίζοντας λογική ανθεκτική σε σφάλματα και περιορισμένη χρήση χώρου — προσέχοντας μάλιστα να μην υπερβούμε το όριο του browser. Με αυτόν τον τρόπο, η εφαρμογή συνδυάζει βέλτιστες πρακτικές για offline–ready λεξικά, γρήγορες αναζητήσεις λέξεων σε μνήμη και ασφαλή, απλή διαχείριση μόνιμων δεδομένων χρήστη, χωρίς επιπλέον server ή cloud εξάρτηση.

5.6 Αλγόριθμος Δημιουργίας Πλέγματος Σταυρολέξου

Η καρδιά της δυναμικής δημιουργίας κάθε επιπέδου σταυρολέξου βρίσκεται στη συνάρτηση **generateCrossword(words, gridSize, pangramSeed)** του αρχείου src/wordGenerator.js. Παρακάτω περιγράγεται αναλυτικά η λογική της.

Πρώτα, από το σύνολο των λέξεων που μπορούν να σχηματιστούν από την τρέχουσα παλέτα γραμμάτων, εντοπίζεται μία λέξη-«σπόρος» (pangram) που περιέχει ακριβώς N διαφορετικά γράμματα (όπου N το μέγεθος της παλέτας). Αυτή η λέξη εγγυάται ότι έχουμε έναν "πυρήνα" με το μέγιστο πλήθος διαφορετικών γραμμάτων, που θα εξασφαλίσει τη μεγιστοποίηση της ποικιλίας. Έπειτα έχουμε την δημιουργία του κενού πλέγματος όπου αρχικοποιείται ένας πίνακας διαστάσεων gridSize × gridSize (τυπικά 12×12), γεμάτος με κενά (empty cells). Σκοπός μας είναι να "**τοιχοθετήσουμε**" τις λέξεις πάνω σε αυτό το πλέγμα, χωρίς επικαλύψεις που συγκρούονται.

Ύστερα από την δημιουργία του κενού πλέγματος πηγαίνουμε στην τοποθέτηση της λέξης-σπόρου. Αφού επιλεχθεί τυχαία αν η pangram θα μπει οριζόντια ("H") ή κάθετα ("V"), σαρώνουμε όλες τις δυνατές θέσεις (κάθε (x,y) τέτοια ώστε να χωρούν όλα τα γράμματα). Μόλις βρεθεί μια θέση όπου ολόκληρη η λέξη χωρά χωρίς να βγει έξω από τα όρια, «βάζουμε» εκεί τη λέξη. Για κάθε μία από τις υπόλοιπες λέξεις, γίνεται σάρωση και δοκιμάζουμε την τοποθέτηση κάθε μίας σε κάθε πιθανό (x,y) τόσο οριζόντια όσο και κάθετα. Σε κάθε κελί που θα καταλάβει η λέξη, αν είναι ήδη καλυμμένο από άλλο γράμμα, το γράμμα πρέπει να συμπίπτει ακριβώς. Σε κάθε θέση που είναι κενή επιτρέπεται και αν βρεθεί σύγκρουση (διαφορετικό γράμμα), η θέση απορρίπτεται. Η τοποθέτηση γίνεται μόλις βρεθεί η πρώτη συμβατή θέση, και η λέξη τοποθετείται εκεί και περνάμε στην επόμενη. Επίσης αν μια λέξη δεν βρει θέση μετά από όλες τις δοκιμές, παραλείπεται (και καταγράφεται ως "extraWord").

Αν μετά από πολλά βήματα δεν τοποθετηθεί επαρκής αριθμός λέξεων ή η pangram αρχικά δεν βρέθηκε, η γεννήτρια μπορεί να ξανακαλέσει τον εαυτό της επιλέγοντας άλλη λέξη-σπόρο ή μειωμένο μέγεθος παλέτας, για να αποφύγει **deadlocks**.

Η generateCrossword επιστρέφει ένα αντικείμενο με δύο κύρια πεδία:

- **words**: πίνακας τοποθετημένων λέξεων, όπου κάθε εγγραφή είναι [x, y, word, direction]
- **extraWords**: πίνακας λέξεων που ήταν δυνατές αλλά δεν χώρεσαν στο πλέγμα.

Με τον τρόπο αυτό, κάθε επίπεδο «ανασχεδιάζεται» από το μηδέν, με νέα παλέτα και τυχαία τοποθέτηση. Η επιλογή pangram ως βάση εξασφαλίζει πλούσια κάλυψη γραμμάτων και μείωση αχρησιμοποίητων κελιών. Το LevelGenerator κρατά ιστορικό (localStorage) ώστε να μην επαναχρησιμοποιούνται ίδια σύνολα γραμμάτων σε διαδοχικά runs

.

5.7 Αποθήκευση και Ιστορικό Επιπέδων

Η εφαρμογή αξιοποιεί το LocalStorage του browser για δύο βασικούς σκοπούς:

Ιστορικό Επιπέδων--Κάθε σύνολο γραμμάτων που έχει ήδη εμφανιστεί στο παιχνίδι μετατρέπεται σε ένα μοναδικό «αποτύπωμα» και αποθηκεύεται σε ένα σύνολο στο LocalStorage. Με αυτόν τον τρόπο, κατά τη δημιουργία νέων επιπέδων ελέγχεται γρήγορα αν η τρέχουσα παλέτα έχει ήδη παιχτεί· αν ναι, αγνοείται και επιλέγεται άλλη, εξασφαλίζοντας ότι ο παίκτης συναντά πάντα φρέσκο περιεχόμενο.

Διατήρηση Προόδου--Το LocalStorage διατηρεί επίσης τις κρίσιμες μεταβλητές της προόδου—όπως ο συνολικός αριθμός πόντων, διαμαντιών και ο τρέχων αριθμός επιπέδου. Έτσι, ακόμη κι αν ο χρήστης κλείσει την εφαρμογή ή ανανεώσει τη σελίδα, η κατάσταση του παιχνιδιού επαναφορτώνεται αυτόματα, επιτρέποντας απρόσκοπτη συνέχεια από εκεί που είχε μείνει.

Επιπλέον, υπάρχει η δυνατότητα «καθαρής εκκίνησης»: κάθε φορά που ξεκινά νέο run, το ιστορικό των παλετών και οι τρέχουσες μετρήσεις διαγράφονται, ώστε το παιχνίδι να προσφέρει μοναδική εμπειρία σε κάθε session.

5.8 Gameplay και Διεπαφή Χρήστη (UI)

Το UI της εφαρμογής έχει σχεδιαστεί πάνω σε Pixi.js, αξιοποιώντας τον καμβά (canvas) για rendering υψηλής απόδοσης και διαδραστικότητα χωρίς καθυστερήσεις. Βασικά στοιχεία του περιβάλλοντος:

Πίνακας Σταυρόλεξου: Δυναμικά προσαρμοζόμενο στο μέγεθος της οθόνης, κεντραρισμένο και με αυτορυθμιζόμενο μέγεθος κελιών ώστε να είναι πάντα ευανάγνωστο.

Παλέτα Γραμμάτων: Γραμμική σειρά από γράμματα που επιλέγει ο παίκτης με drag-and-drop για να σχηματίσει λέξεις.

Πάνελ Βοηθειών: Κουμπιά για “Hint” (αποκάλυψη γράμματος), special highlight με χρονοδιακόπτη και διαμάντια, καθώς και ένα i-κουμπί που ανοίγει παράθυρο με τους κανόνες. Κάθε κουμπί ενημερώνει άμεσα την οθόνη με pop-ups και αλληλεπιδράσεις.

Αντίστροφος Χρονοδιακόπτης & Highlights: Κόκκινα πλαίσια γύρω από τυχαία κελιά, μικρός κυκλικός μετρητής χρόνου που εμφανίζεται επάνω δεξιά και μειώνεται γραφικά με έναν δακτύλιο GSAP στο canvas.

Popup Μηνύματα: Όλες οι ενημερώσεις πόντων, διαμαντιών, σφαλμάτων (π.χ. αν δεν επαρκούν πόντοι) ή ολοκλήρωσης επιπέδου εμφανίζονται ως διακριτικά μηνύματα στο κέντρο της οθόνης, διατηρώντας τη ροή του παιχνιδιού άμεση και ευχάριστη.

Αυτή η αρχιτεκτονική UI, σε συνδυασμό με το έξυπνο ιστορικό επιπέδων και τη διατήρηση προόδου, εξασφαλίζει μια συνεπή, γρήγορη και καθηλωτική εμπειρία που εξελίσσεται ομαλά όσο ο παίκτης προχωρά σε όλο και δυσκολότερους γρίφους.

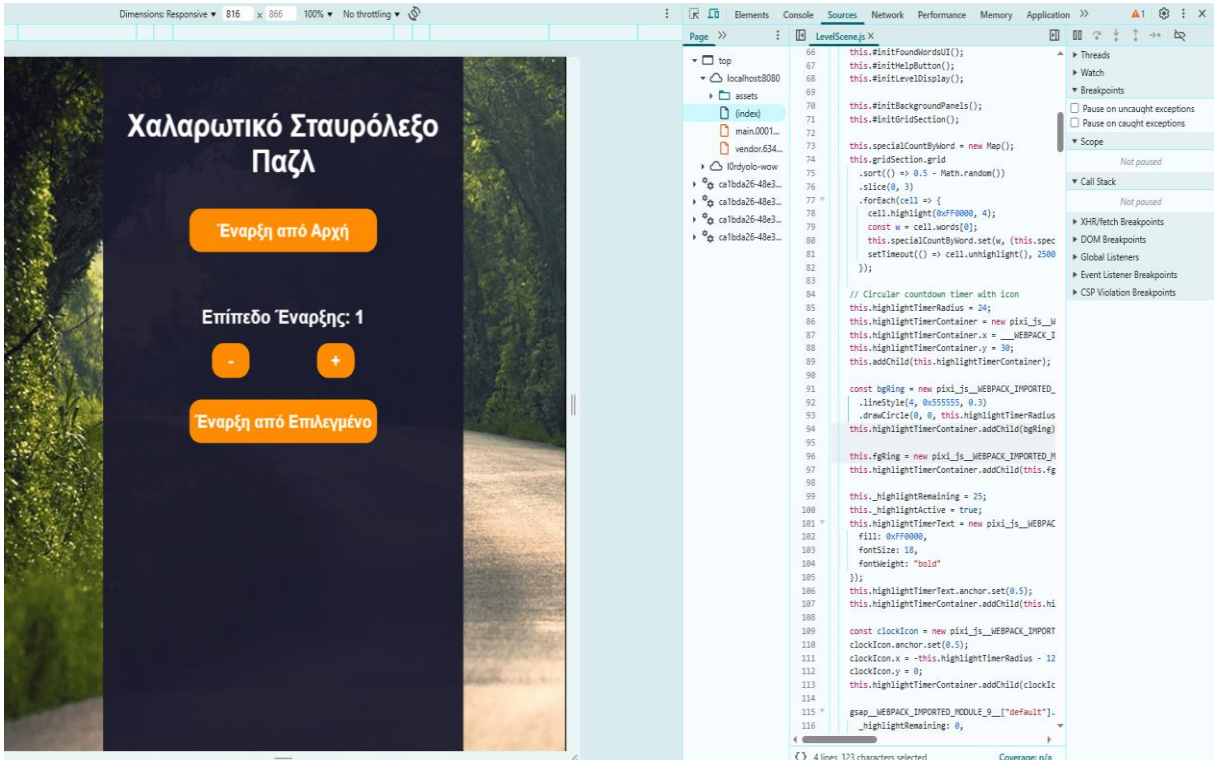
Κεφάλαιο 6ο: Debug & Live Inspection με DevTools

Στο επόμενο κεφάλαιο θα χρησιμοποιήσουμε τα εργαλεία ανάπτυξης (DevTools) του browser προκειμένου να δούμε “ζωντανά” τον εσωτερικό κόσμο του παιχνιδιού. Αντί απλώς να διαβάζουμε θεωρία για το πώς λειτουργεί το πλέγμα, οι λίστες λέξεων ή ο μηχανισμός αποκαλύψεων, θα φορτώσουμε το LevelScene στο πρόγραμμα περιήγησης και θα εκτελέσουμε μερικές απλές εντολές – live queries μέσα στη **Console**.

Μέσα από αυτά τα παραδείγματα θα:

- Εξετάσουμε τη δομή των Grid Cells και θα κατανοήσουμε πώς αποθηκεύονται στη μνήμη.
- Επισκοπήσουμε σε πραγματικό χρόνο ποιες λέξεις έχει ήδη βρει ο παίκτης.
- Δοκιμάσουμε δικές μας “παλέτες” γραμμάτων και θα δούμε αμέσως πόσες λέξεις σχηματίζονται.

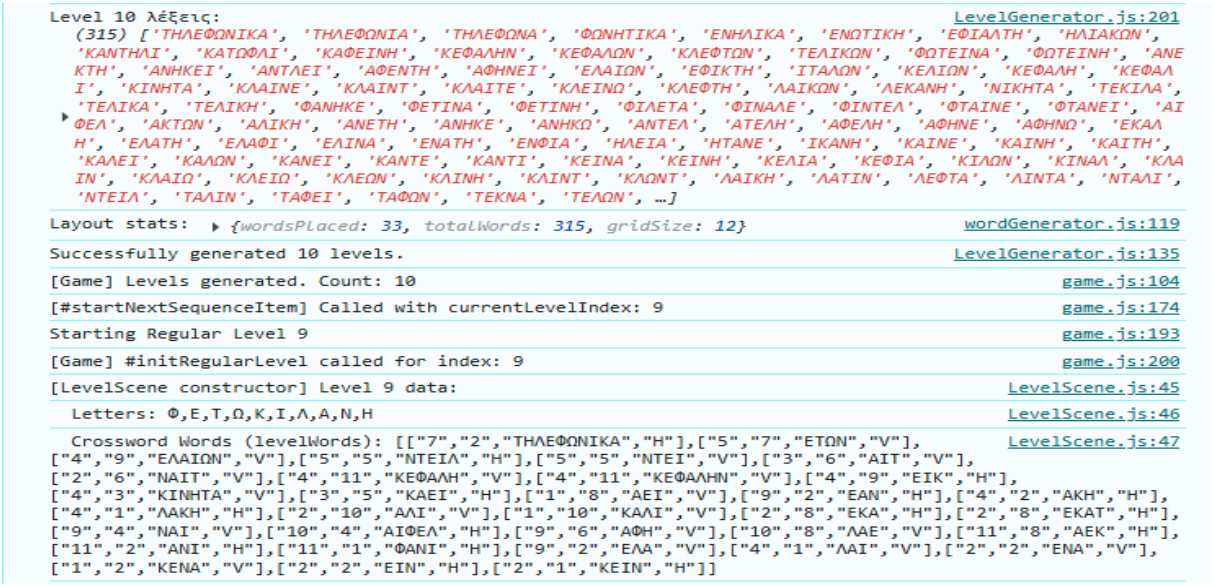
Αυτή η hands-on προσέγγιση βοηθάει όχι μόνο να ελέγξουμε ότι ο κώδικας λειτουργεί σωστά, αλλά και να κατανοήσουμε σε βάθος τη λογική του παιχνιδιού, να εντοπίσουμε γρήγορα πιθανά σφάλματα και τελικά να βελτιώσουμε την εμπειρία του χρήστη.



Εικόνα 49: Αρχικό μενού και Dev Tools

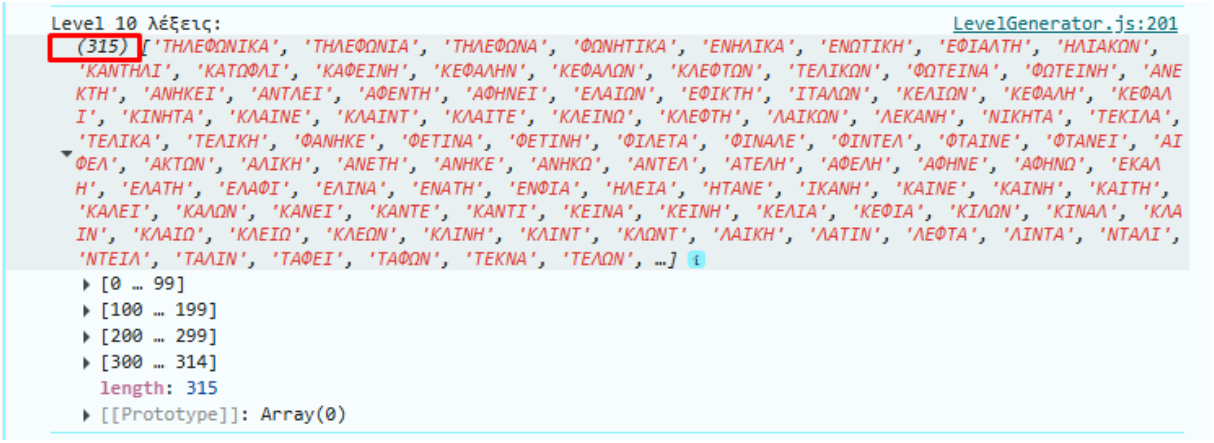
6.1 Επισκόπηση Λιστών Λέξεων με DevTools

Θα μεταβούμε στο επίπεδο 10 του παιχνιδιού και θα δούμε πώς μπορούμε να φέρουμε στην επιφάνεια τις εσωτερικές λίστες λέξεων και να καταλάβουμε πώς «βλέπει» ο κώδικάς μας το επίπεδο.Αφού ανοίξουμε την κονσόλα μπορούμε να τρέξουμε πολλά παραδείγματα αφού πρώτα όμως έχουμε εκχωρήσει στο τρέχον LevelScene ένα global αντικείμενο με την γραμμή window.currentLevelScene.Έτσι μπορούμε να πάρουμε λέξεις που απομένουν στο πλέγμα (Remaining Words), επικυρωμένες εκτός πλέγματος λέξεις (Extra Words), ακόμα και αυτές που έχετε ήδη βρει δηλαδή τις βασικές λέξεις.



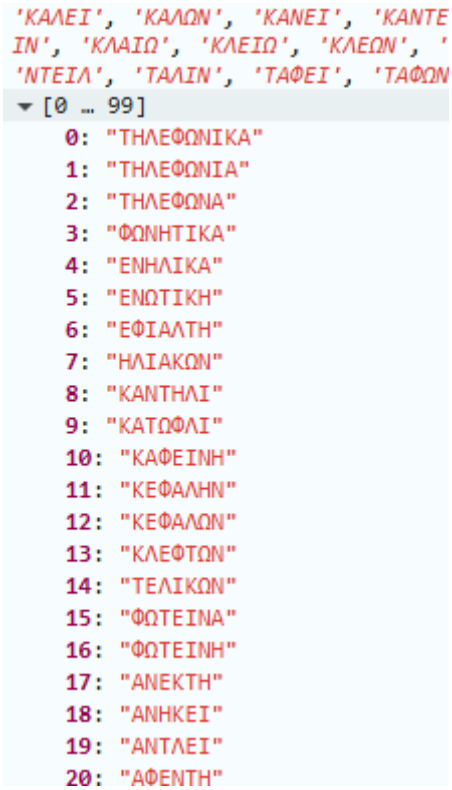
Εικόνα 50: Επίπεδο 10 μέσω Dev Tools

Στην παραπάνω εικόνα βλέπουμε ένα τυπικό log από την κονσόλα κατά τη στιγμή που το παιχνίδι δημιουργεί και ξεκινά το Επίπεδο 10.



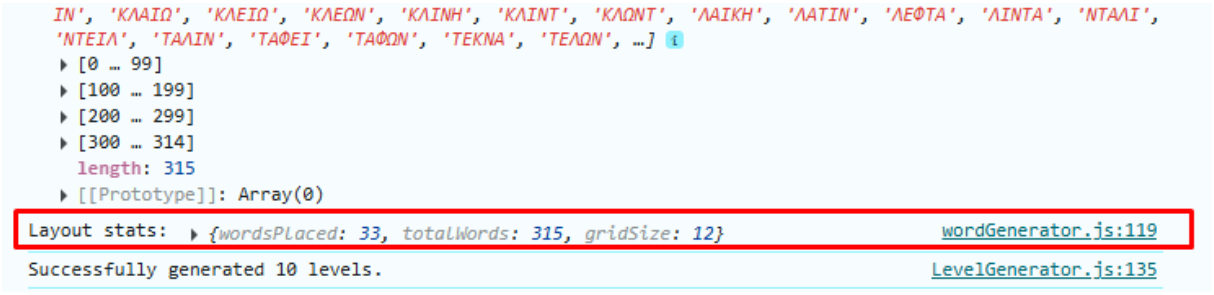
Εικόνα 51: Σύνολο λέξεων

Εδώ έχει φιλτραριστεί ένα σύνολο 315 λέξεων από το λεξικό, όλες οι πιθανές λέξεις που μπορούν να σχηματιστούν με την παλέτα γραμμάτων του επιπέδου. Αν πατήσουμε στα βελάκια του κάθε πίνακα από κάτω θα ανοίξουν όλες οι λέξεις αριθμημένες.



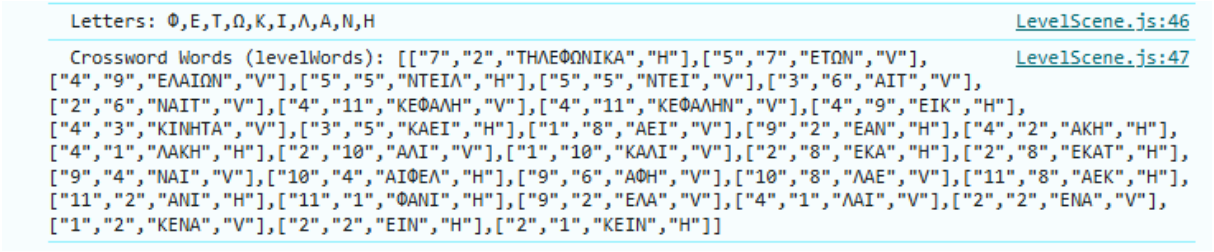
Εικόνα 52: Λέξεις Πίνακα

Μπορούμε να δούμε το μέγεθος του πλέγματος καθώς και τις λέξεις που βρέθηκαν και τοποθετήθηκαν μέσω του πεδίου Layout stats.



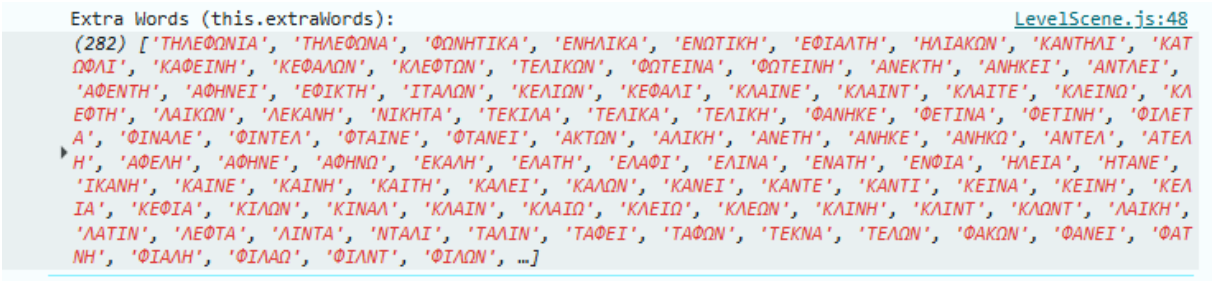
Εικόνα 53: GridSize και TotalWords

Στην παραπάνω γραμμή καταλαβαίνουμε το σύνολο των λέξεων που είναι 315, πόσες από αυτές τις λέξεις τελικά μπήκαν στο σταυρόλεξο (33),και το μέγεθος του πλέγματος (12×12).Άν προχωρήσουμε πιο κάτω θα δούμε τα γράμματα της παλέτας καθώς και τις συντεταγμένες της κάθε λέξης.



Εικόνα 54: Συντεταγμένες λέξεων

Όπως έχουμε πει και στο προηγούμενο κεφάλαιο ο πίνακας λέξεων τοποθετεί τις λέξεις με συντεταγμένες (x, y). Η μορφή όπως βλέπουμε πιο πάνω είναι οι συντεταγμένες x, y, η λέξη και η κατεύθυνση (‘H’ οριζόντια, ‘V’ κάθετα).Έπειτα από τις βασικές λέξεις και τις συντεταγμένες τους υπάρχουν πίνακες όπως των βασικών λέξεων, με τις έξτρα λέξεις με την ίδια λογική.



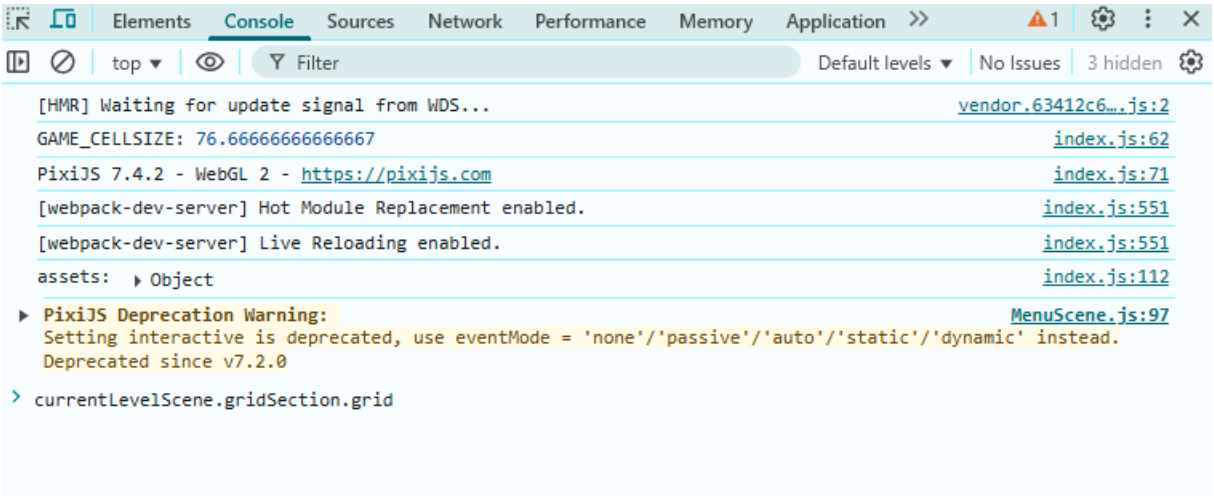
Εικόνα 55: Πίνακας Extra Λέξεων

6.2 Πρόσβαση στην Κατάσταση των Grid Cells

Στην καρτέλα **Console** των DevTools μπορούμε να «ξεκλειδώσουμε» την εσωτερική δομή του παιχνιδιού και να δούμε κάθε κελί (Grid Cell) που αποτελεί το πλέγμα της λέξης-σταυρόλεξο. Κάθε κελί περιέχει τέσσερις βασικές πληροφορίες:

- **Θέση του κελιού** (συντεταγμένες x,y) για να καταλάβουμε σε ποιο σημείο του πλέγματος ανήκει.
- **Γράμμα που εμφανίζει**, το οποίο αποθηκεύεται στην ιδιότητα letter του κελιού.
- **Κατάσταση “ξεκλειδώματος”** (unlocked ή locked), που μας δείχνει αν ο παίκτης το έχει ήδη αποκαλύψει.
- **Λίστα λέξεων** (words) στις οποίες συμμετέχει το γράμμα αυτό, χρήσιμη για να εντοπίζουμε πού τέμνονται οι λέξεις.

Στο constructor του LevelScene έχουμε κάνει window.currentLevelScene = this έτσι μπορούμε πηγαίνοντας στο Console μπορούμε να γράψουμε currentLevelScene.gridSection.grid για να δούμε τον πίνακα όλων των κελιών και τις ιδιότητες τους π.χ. θέση, γράμμα, κ.λπ. Αφού επιλέξουμε το επίπεδο που θέλουμε ώστε να ξεκινήσει το παιχνίδι, ανοίγουμε το Console και βλέπουμε του πίνακες με τις λέξεις.

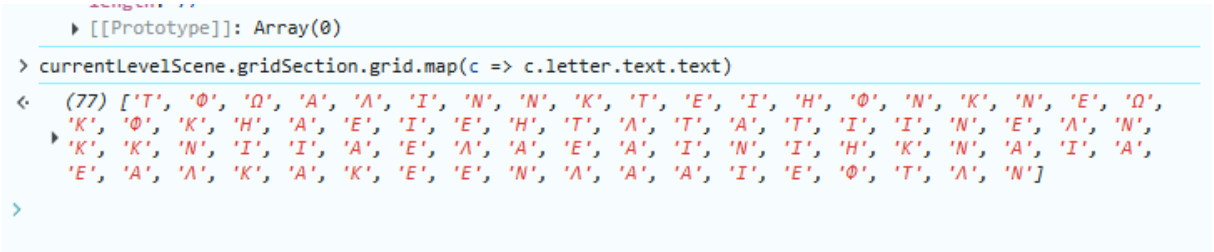


Εικόνα 56: Console και ξεκλείδωμα δομής

Μόλις πατήσουμε Enter τότε μπορούμε να έχουμε πρόσβαση στον πίνακα με όλα τα κελιά αντικείμενα. Για κάθε κελί μπορούμε να ελέγξουμε την θέση το γράμμα και την λίστα που ανήκει.

Εικόνα 57: Ξεκλείδωμα κελιών

Αν θέλουμε να δούμε μόνο τα γράμματα όλων των κελιών, στο Console δίνουμε την γραμμή:

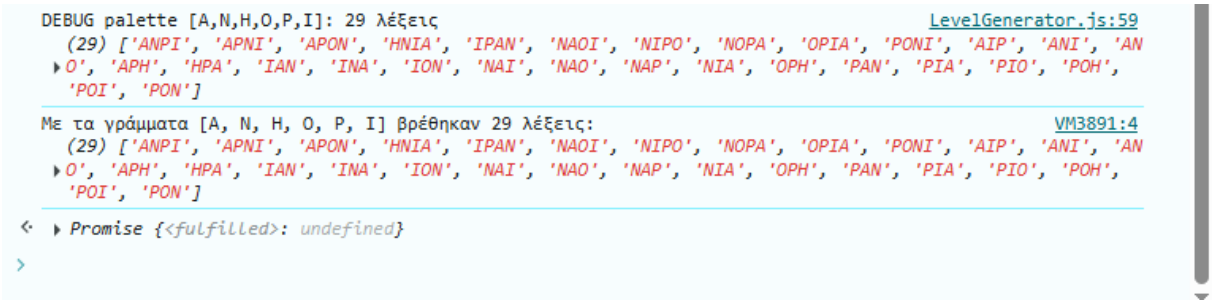


Εικόνα 58: Γράμματα όλων των κελιών

6.3 Δοκιμή Χειροκίνητης Παλέτας

Για να κατανοήσουμε καλύτερα πώς το παιχνίδι επιλέγει και δημιουργεί τις λέξεις από μια συγκεκριμένη ομάδα γραμμάτων, μπορούμε να πραγματοποιήσουμε μια «χειροκίνητη» δοκιμή παλέτας μέσω των DevTools του προγράμματος περιήγησης. Για να εξερευνήσουμε ζωντανά τη λογική δημιουργίας των σταυρολέξων, ανοίγουμε πρώτα τα DevTools του προγράμματος περιήγησης (πατώντας F12 ή με δεξί κλικ → «Inspect»). Στο παράθυρο που ανοίγει, μεταφερόμαστε στην καρτέλα **Console**, όπου μπορούμε να εκτελέσουμε εντολές JavaScript μέσα στο περιβάλλον του παιχνιδιού.Προκειμένου να έχουμε άμεση πρόσβαση στις εσωτερικές μεθόδους του παιχνιδιού, στον constructor της κλάσης **LevelScene** ορίζουμε window.currentLevelScene = this. Έτσι, από την κονσόλα μπορούμε απλώς να γράψουμε currentLevelScene και να καλέσουμε οποιαδήποτε βοηθητική συνάρτηση του επιπέδου, όπως για παράδειγμα τη debugWithPalette([...]) για να δοκιμάσουμε χειροκίνητες παλέτες γραμμάτων. Με αυτόν τον τρόπο βλέπουμε σε πραγματικό χρόνο πόσες και ποιες λέξεις «βγάζει» το παιχνίδι για οποιονδήποτε συνδυασμό γραμμάτων επιλέξουμε. Ας υποθέσουμε ότι θέλουμε να δοκιμάσουμε την ομάδα γραμμάτων ['A','N','H', 'O','P','I'].

Αφού πρώτα έχουμε προσθέσει στον constructor του **Game** την γραμμή window.levelGen = this.levelGenerator, αποκτούμε πρόσβαση στον generator και στο τρέχον επίπεδο από την κονσόλα.



Εικόνα 59: Δοκιμή Χειροκίνητης Παλέτας

Η debugWithPalette() φροντίζει μόνη της να φορτώσει το λεξικό (αν δεν έχει ήδη συμβεί), να καλέσει τη συνάρτηση generateWords() και να επιστρέψει τη λίστα λέξεων.

Στο παράδειγμα που φαίνεται στην εικόνα καλέστηκε η παλέτα **[A, N, H, O, P, I,]** και η μέθοδος επέστρεψε **68 λέξεις**, τις οποίες εμφάνισε τόσο η LevelGenerator (γραμμή 59) όσο και η κλήση από την κονσόλα (VM3880:4).

Με αυτό το Live Inspection βλέπουμε άμεσα ποια και πόσα λήμματα του λεξικού σχηματίζονται από ένα αυθαίρετο σύνολο γραμμάτων, χωρίς να χρειαστεί να τροποποιήσουμε τον βασικό κώδικα του παιχνιδιού. Αυτό βοηθάει στον έλεγχο της ποιότητας του λεξικού, στον πειραματισμό με νέες παλέτες και στην άμεση διάγνωση τυχόν ελλείψεων ή προβλημάτων.

6.4 Τροποποίηση LocalStorage

Στην καρτέλα **Application ➔ Local Storage** των DevTools μπορούμε να δούμε και να επεξεργαστούμε άμεσα τα δεδομένα που αποθηκεύει το παιχνίδι στο πρόγραμμα περιήγησης. Στην εικόνα βλέπουμε:

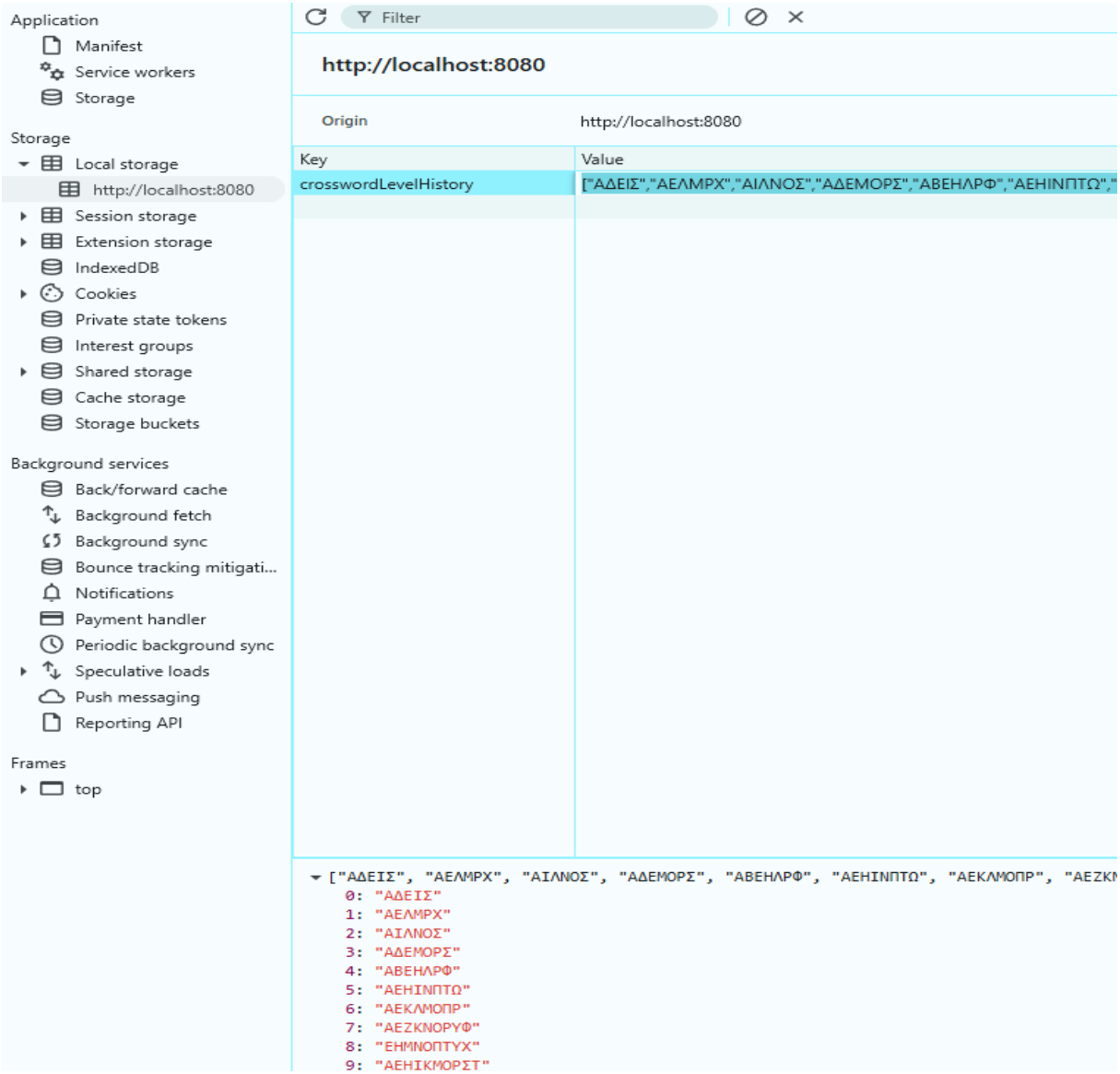
- **Origin**: Το URL της τοποθεσίας όπου τρέχει το παιχνίδι (εδώ http://localhost:8080).
- **Key**: crosswordLevelHistory
  Κρατάει έναν πίνακα από «παλέτες» γραμμάτων (arrays με τα γράμματα κάθε επιπέδου) που έχουν ήδη χρησιμοποιηθεί.
- **Value**: Ένα JSON string με τις παλέτες, π.χ.
  ["ΑΔΕΙΣ","ΑΕΛΜΡΧ","ΑΙΛΝΟΣ","ΑΔΕΜΟΡΣ", …]

Κάθε στοιχείο είναι η σειρά γραμμάτων ενός επιπέδου, κωδικοποιημένη ως απλό string.Εδώ μπορούμε να επιλέξουμε το κλειδί crosswordLevelHistory για να δούμε ποιες παλέτες έχουν ήδη εμφανιστεί στον τρέχοντα κύκλο του παιχνιδιού. Πατώντας δεξί κλικ στο crosswordLevelHistory και έπειτα **Delete.**

Μπορούμε να επεξεργαστούμε χειροκίνητα το JSON string, να προσθέσουμε ή να αφαιρέσουμε παλέτες, και μετά να κάνουμε **Refresh** στο παιχνίδι:

- Αν αφαιρέσουμε όλα τα στοιχεία, το επίπεδο 1 θα φορτώσει και πάλι από την αρχή με νέα, διαφορετική παλέτα.
- Αν προσθέσουμε μία «νέα» παλέτα, ο generator θα περάσει κατευθείαν στο επόμενο επίπεδο, παρακάμπτοντας το τρέχον.

Με αυτό τον τρόπο αποδεικνύεται ζωντανά πώς το παιχνίδι χρησιμοποιεί το localStorage για να διασφαλίσει ότι κάθε σετ γραμμάτων εμφανίζεται μόνο μία φορά ανά run, αλλά και πώς μπορούμε να πειραματιστούμε άμεσα με τα δεδομένα για debugging ή demo.



Εικόνα 60: Τροποποίηση LocalStorage

6.5 Συμπέρασμα & Επόμενα Βήματα

Με τη χρήση των εργαλείων ανάπτυξης (DevTools) του browser, αποκτούμε άμεση πρόσβαση στον "εσωτερικό κόσμο" του παιχνιδιού μας, χωρίς να χρειάζεται να τροποποιούμε τον κώδικα ή να επανεκκινούμε την εφαρμογή. Μέσα από ζωντανές εξετάσεις και πειραματισμό:

- **Επισκόπηση Λιστών Λέξεων**-- Βλέπουμε σε πραγματικό χρόνο πόσες και ποιες λέξεις "βγάζει" ο LevelGenerator για κάθε επίπεδο, κατανοώντας καλύτερα τα κριτήρια επιλογής και φιλτραρίσματος.
- **Κατάσταση Grid Cells**-- Επιθεωρούμε άμεσα κάθε κελί του πλέγματος, μαθαίνοντας ποια γράμματα είναι ξεκλείδωτα, σε ποιες λέξεις ανήκουν και πού τέμνονται.

- **Δοκιμή Χειροκίνητης Παλέτας--** πειραματιζόμαστε με οποιονδήποτε συνδυασμό γραμμάτων για να αξιολογήσουμε την πληρότητα του λεξικού μας και τη συμπεριφορά του μηχανισμού generateWords().
- **Τροποποίηση LocalStorage--** Παρεμβαίνουμε άμεσα στο crosswordLevelHistory για να ελέγξουμε πώς ο generator αποφεύγει τις επαναλήψεις και να αναγκάσουμε το παιχνίδι να φορτώσει νέες (ή ήδη χρησιμοποιημένες) παλέτες.

Αυτή η hands-on προσέγγιση δεν εξυπηρετεί μόνο τον εντοπισμό σφαλμάτων και την επαλήθευση της ορθότητας του κώδικα, αλλά προσφέρει και βαθειά κατανόηση της λογικής του παιχνιδιού. Με τον τρόπο αυτό:

1. **Επιταχύνουμε τον Εντοπισμό Σφαλμάτων:** Ανιχνεύουμε άμεσα τα σημεία όπου η ροή του προγράμματος αποκλίνει από τις προσδοκίες μας.
2. **Βελτιώνουμε την Ποιότητα του Λεξικού:** Εντοπίζουμε γρήγορα κενά ή υπερβολικές λέξεις μέσα στα αποτελέσματα του generateWords().
3. **Δοκιμάζουμε Νέες Ιδέες σε Πραγματικό Χρόνο:** Πειραματιζόμαστε με ό,τι θέλουμε — γραμματικές παλέτες, κανόνες hints, τιμές στο localStorage — και βλέπουμε αμέσως το αποτέλεσμα.

Στο επόμενο στάδιο, θα μπορούσε να:

- Υπάρξουν **breakpoints** και να γίνουν **step-through** σε συναρτήσεις κλειδιά, όπως η #CometLetterHint() ή το generateLevel(), ώστε να κατανοηθεί κάθε βήμα εκτέλεσης.
- Χρησιμοποιηθεί την καρτέλα **Network** για να παρατηρηθούν τυχόν αιτήσεις (π.χ. φόρτωση λεξικού) και τις αποκρίσεις τους.
- Δοκιμή **ζωντανής τροποποίησης στυλ** μέσα στην καρτέλα Elements για να φανεί πώς αλλάζουν άμεσα τα χρώματα, τα μεγέθη και η διάταξη των UI στοιχείων.

Με αυτά τα εργαλεία στη διάθεσή μας, η ανάπτυξη, ο έλεγχος και η βελτίωση του παιχνιδιού γίνονται πολύ πιο αποδοτικά, διαδραστικά και — κυρίως — κατανοητά.

Κεφάλαιο 7ο: Οδηγίες Εγκατάστασης και Εκτέλεσης

7.1 Οδηγίες Εγκατάστασης και Εκτέλεσης

Η εφαρμογή έχει υλοποιηθεί αποκλειστικά με τεχνολογίες ιστού και έχει σχεδιαστεί ώστε να εκτελείται με ευκολία σε οποιοδήποτε σύγχρονο σύστημα χωρίς την ανάγκη εγκατάστασης. Η ευελιξία αυτή επιτυγχάνεται χάρη στην πλήρη αξιοποίηση της JavaScript και του rendering μέσω καμβά, καθιστώντας το παιχνίδι ιδιαίτερα ελαφρύ και προσβάσιμο.

1) Για να τρέξετε το παιχνίδι τοπικά στον υπολογιστή σας, ακολουθήστε τα παρακάτω βήματα:

2) Βεβαιωθείτε ότι έχετε εγκατεστημένο το Node.js (έκδοση LTS ή νεότερη), το οποίο περιλαμβάνει τον npm.

3) Αν δεν έχετε ήδη το Git, εγκαταστήστε το για να κάνετε clone το αποθετήριο.

4) Κάντε clone το αποθετήριο του παιχνιδιού με την εντολή git clone ακολουθούμενη από το URL του project.

5) Μπείτε στο φάκελο του project με cd relaxing-crossword (ή το όνομα του φακέλου όπως δημιουργήθηκε).

6) Εκτελέστε npm install για να κατεβάσετε και να εγκαταστήσετε όλες τις βιβλιοθήκες που απαιτούνται (Pixi.js, GSAP, wordfreq κ.λπ.).

7) Η διαδικασία αυτή δημιουργεί έναν φάκελο node_modules με ολόκληρο το περιβάλλον εκτέλεσης.

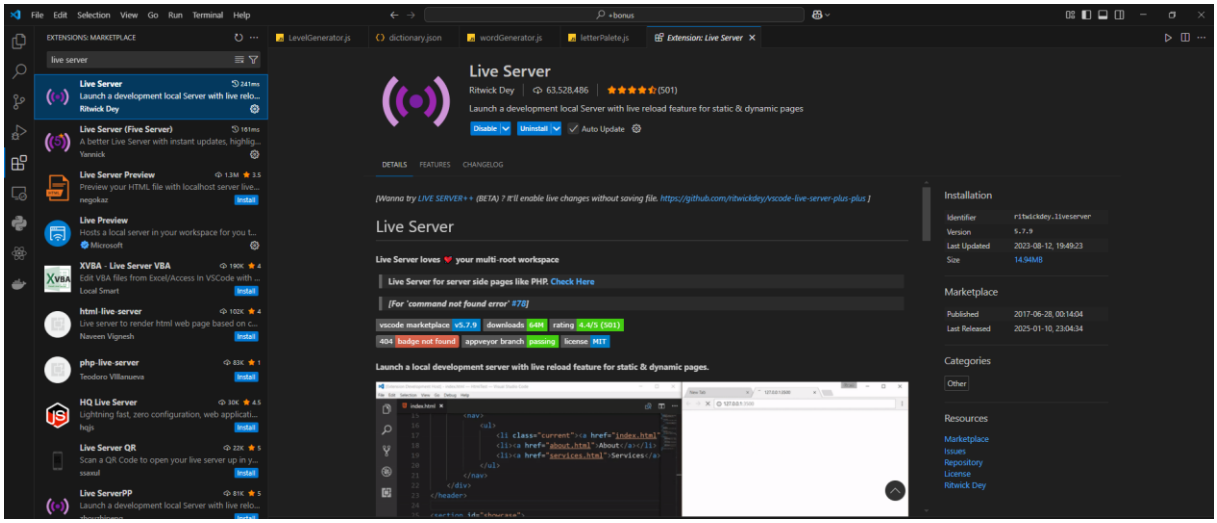
8) Μετά την επιτυχή εγκατάσταση, ξεκινήστε τον τοπικό server ανάπτυξης με npm run dev ή την αντίστοιχη εντολή που έχει οριστεί στο package.json.

9) Ανοίξτε τον browser στη διεύθυνση http://localhost:8080 (ή τη θύρα που εμφανίζεται στο τερματικό) για να δείτε και να δοκιμάσετε το παιχνίδι.

10) Για να ετοιμάσετε στατικά αρχεία προς διάθεση σε Production, τρέξτε npm run build.Τα παραγόμενα αρχεία θα βρεθούν σε φάκελο dist ή build, έτοιμα για ανάρτηση σε web server.

Οι παραπάνω οδηγίες έχουν να κάνουν ανεξαρτήτως IDE.Στο VS Code ακολουθούν τα βήματα για την εγκατάσταση:

4. Κλωνοποιούμε ή κατεβάζουμε το αποθετήριο του project σε έναν τοπικό φάκελο.
5. Ανοίγουμε τον φάκελο με το Visual Studio Code
6. Εγκαθιστούμε το extension **Live Server**.
7. Κάνουμε δεξί κλικ στο αρχείο index.html και επιλέγουμε **Open with Live Server**.



Εικόνα 61: Επιλογή Live Server από το μενού στο VS Code

Εναλλακτικά, αν χρησιμοποιείται Node.js, μπορεί να γίνει εκτέλεση μέσω του http-server:

```
npm install -g http-server
http-server
```

Η εφαρμογή ανοίγει αυτόματα στον προεπιλεγμένο browser στη διεύθυνση http://localhost:8080.

7.2 Βασικές απαιτήσεις συστήματος

Για την εγκατάσταση και εκτέλεση της εφαρμογής απαιτείται:

- Ένας μοντέρνος browser με υποστήριξη WebGL (Chrome, Firefox, Edge)
- Ένα πρόγραμμα τοπικού εξυπηρετητή (όπως το Live Server στο Visual Studio Code ή το http-server της Node.js)
- Ελάχιστες γνώσεις χειρισμού αρχείων HTML και JavaScript

7.3 Έλεγχος λειτουργίας

Μόλις εκκινηθεί η εφαρμογή, εμφανίζεται η αρχική οθόνη με το πρώτο επίπεδο του σταυρόλεξου. Η εμφάνιση αυτής της σκηνής επιβεβαιώνει την επιτυχία της εκτέλεσης.

Σε περίπτωση που η εφαρμογή δεν φορτώνει σωστά:

- Επιβεβαιώνουμε ότι το dictionary.json βρίσκεται στον ίδιο φάκελο με τα υπόλοιπα αρχεία
- Βεβαιωνόμαστε ότι τα αρχεία σερβίρονται μέσω HTTP/localhost και όχι ως τοπικά (file://)
- Ελέγχουμε αν ο browser υποστηρίζει WebGL

Η responsive σχεδίαση της εφαρμογής επιτρέπει την εκτέλεση της και σε κινητές συσκευές, χωρίς απώλεια λειτουργικότητας. Το interface προσαρμόζεται δυναμικά στις οθόνες tablets και smartphones. Οι χρήστες μπορούν να αλληλεπιδρούν με τα γράμματα και το grid μέσω αφής, όπως ακριβώς με το ποντίκι σε κινητό και tablet. Το παιχνίδι εκτελείται σε tablet σε portrait διάταξη.

7.4 Προτάσεις διανομής και φιλοξενίας

Η εφαρμογή μπορεί να διανεμηθεί με διάφορους τρόπους:

- Μέσω **GitHub Pages**, για online πρόσβαση χωρίς εγκατάσταση
- Φιλοξενία σε εκπαιδευτικό intranet
- Εγκατάσταση σε υπολογιστές εργαστηρίων για offline χρήση

Η πλήρης ανεξαρτησία από back-end server και η ελαφριά φύση του κώδικα επιτρέπουν την αξιοποίησή του σε πλήθος εκπαιδευτικών και μη περιβαλλόντων, καθιστώντας την εφαρμογή ιδανική για ευρεία χρήση.

Κεφάλαιο 8ο: Παραμετροποίηση και Επεκτασιμότητα

8.1 Διαχείριση Λεξικού

Η βιβλιοθήκη wordfreq της Python χρησιμοποιείται στο αρχείο get_words.py για την επεξεργασία των ελληνικών λέξεων. Πρόκειται για ένα ισχυρό εργαλείο που παρέχει συχνότητες λέξεων από μεγάλες βάσεις δεδομένων, βοηθώντας έτσι στον φιλτράρισμα των πιο κοινών και σημαντικών λέξεων του λεξικού. Χρησιμοποιείται για:

- Καθαρισμό σπάνιων ή άγνωστων λέξεων
- Διατήρηση μόνο όσων έχουν λεξική ή παιγνιώδη αξία
- Κανονικοποίηση μορφής (π.χ. αφαίρεση τόνων)

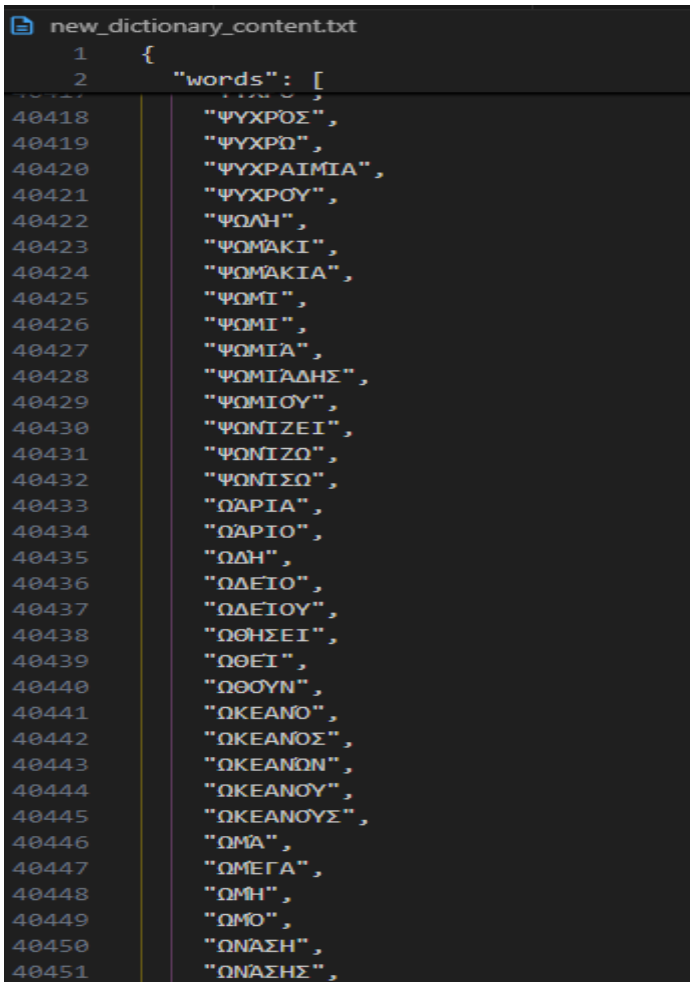
Αυτό καθιστά το λεξικό πιο χρήσιμο και κατάλληλο για παικτική χρήση χωρίς λέξεις που δεν αναγνωρίζει εύκολα ένας μέσος χρήστης. Η βασική πηγή δεδομένων του παιχνιδιού είναι το αρχείο dictionary.json, το οποίο δημιουργείται από το script get_words.py. Το αρχείο αυτό περιλαμβάνει τις λέξεις που θεωρούνται έγκυρες και φορτώνεται ασύγχρονα στην εφαρμογή κατά την εκκίνηση μέσω της συνάρτησης loadDictionary().

Η επεξεργασία του λεξικού επιτρέπει:

- Προσθήκη νέων λέξεων
- Αφαίρεση ή αντικατάσταση υπαρχουσών
- Υλοποίηση θεματικών λεξικών (π.χ. γεωγραφία, επιστήμες, αγγλισμοί)

Η τροποποίηση μπορεί να γίνει μέσω απλού text editor ή εργαλείου JSON editor. Το αρχείο dictionary.json παράγεται μέσω του get_words.py, το οποίο χρησιμοποιεί τη βιβλιοθήκη wordfreq για τον καθαρισμό και φιλτράρισμα λέξεων.

Το αρχείο dictionary.json ακολουθεί μια απλή, ευανάγνωστη δομή JSON, ώστε να μπορεί να φορτωθεί εύκολα από την εφαρμογή και να ενημερωθεί όποτε χρειάζεται. Η βασική μορφή είναι η εξής:



Εικόνα 62: Το αρχείο new_dictionary_content.txt ανοιχτό σε επεξεργασία

Το συγκεκριμένο τμήμα κώδικα είναι υπεύθυνο για τη δημιουργία του τελικού αρχείου dictionary.json, που περιλαμβάνει όλες τις λέξεις του λεξικού σε μορφή κατάλληλη για χρήση από την εφαρμογή. Ο κώδικας:

- Εξασφαλίζει σωστό escape χαρακτήρων (π.χ. quotes)
- Δημιουργεί μία σωστά δομημένη λίστα JSON ("words": [])
- Εκτυπώνει επιβεβαίωση στο τέλος με πληροφορίες για την έξοδο

Η έξοδος αυτού του script χρησιμοποιείται αργότερα από τον client για την επαλήθευση των λέξεων κατά τη διάρκεια του gameplay.

Το αρχείο get_words.py μπορεί να εξελιχθεί περαιτέρω ώστε να υποστηρίζει επιλογές μέσω παραμέτρων, όπως:

- Φιλτράρισμα με βάση τη συχνότητα (min_freq)
- Εξαγωγή θεματικών κατηγοριών
- Ανάλυση με βάση το πλήθος χαρακτήρων

Μπορούν να προστεθούν CLI παράμετροι (π.χ. με argparse) ώστε το script να είναι ευέλικτο και επαναχρησιμοποιήσιμο για πολλά λεξικά.

8.2 Πρόσθετα UI/UX Features για Μελλοντική Επέκταση

Για να διατηρήσουμε την εφαρμογή μας ευέλικτη και έτοιμη να εξελιχθεί με τις μελλοντικές απαιτήσεις των χρηστών, προτείνουμε τις παρακάτω βελτιώσεις στο επίπεδο του interface και της εμπειρίας χρήστη (UI/UX):

Dark Mode / Theme Support

Η υποστήριξη πολλαπλών θεμάτων (“themes”) επιτρέπει στον παίκτη να επιλέγει ανάμεσα σε φωτεινή και σκοτεινή εμφάνιση, ανάλογα με τις συνθήκες φωτισμού ή τις προσωπικές του προτιμήσεις.

1) Στο Pixi.js, μπορούμε να ορίσουμε μεταβλητές χρωμάτων (π.χ. **--background-color**, **--cell-border-color**, **--text-color**) και να τις εφαρμόζουμε δυναμικά στις Graphics και Text περιπτώσεις.

2) Ένα toggle στο μενού ρυθμίσεων θα αλλάζει αμέσως τις τιμές αυτές, χωρίς επανεκκίνηση.

3) Η διατήρηση της επιλογής μπορεί να αποθηκεύεται στο localStorage, ώστε σε κάθε νέο παιχνίδι να εφαρμόζεται αυτόματα.

Accessibility (A11y)

Η προσβασιμότητα εξασφαλίζει ότι το παιχνίδι μπορεί να παιχτεί από όσο το δυνατόν ευρύτερο κοινό, συμπεριλαμβανομένων ατόμων με μειωμένη όραση ή κινητικές δυσκολίες.

1) Keyboard Navigation: υποστήριξη επιλογής γραμμάτων και υποβολής λέξης μέσω πλήκτρων (βέλη, Enter, Space).

2) Screen-reader Labels: προσθήκη περιγραφικού aria-label σε όλα τα διαδραστικά στοιχεία (κουμπιά, πίνακες λέξεων, κ.λπ.).

3) Contrast Ratios: έλεγχος των χρωματικών συνδυασμών ώστε να πληρούνται οι οδηγίες WCAG 2.1 (π.χ. λευκός τίτλος σε σκούρο φόντο με τουλάχιστον 4.5:1 αναλογία).

Custom Animations & Transitions

Οι ομαλές κινήσεις και οι «οπτικές ροές» αυξάνουν την αίσθηση ποιότητας και «ζωντάνιας» στο παιχνίδι.

1) Framer-motion Hooks (στα React-based UI panels) για fade-in/out των pop-ups, slide των πινάκων λέξεων, και bounce effects στα γράμματα όταν ξεκλειδώνουν.

2) shadcn/ui Components για εμφάνιση καρτών με εικονίδια και transitions κατά την εναλλαγή επιπέδων ή κατά την προσαρμογή ρυθμίσεων.

Responsive & Mobile-First

Η σωστή κλιμάκωση εξασφαλίζει άνετο χειρισμό σε οθόνες όλων των μεγεθών, από desktop μέχρι tablet και κινητά.

1) Δυναμικός υπολογισμός cellSize βάσει window.innerWidth και innerHeight, με όριο ελάχιστου μεγέθους (π.χ. 20px) για να διατηρούνται αναγνώσιμα τα γράμματα.

2) Υποστήριξη **touch events** για σύρσιμο γραμμάτων στην παλέτα και pinch-to-zoom αν χρειαστεί σε πολύ μεγάλα grids.

Multiplayer / Leaderboards

Για να δώσουμε νέο κοινωνικό χαρακτήρα στο παιχνίδι, μπορούμε να ενσωματώσουμε:

1) WebSocket Lobby: δίκτυο σε πραγματικό χρόνο όπου δύο ή περισσότεροι παίκτες ανταγωνίζονται στο ίδιο grid, εμφανίζοντας live ποιος ολοκληρώνει πρώτος.

2) Firebase Realtime Database ή Supabase για αποθήκευση σκορ, εμφάνιση global leaderboards, και ιστορικό προσωπικών επιδόσεων.

Analytics & Telemetry

Η συλλογή δεδομένων χρήσης βοηθά στον εντοπισμό δημοφιλών επιπέδων, συχνότητας χρήσης βοηθειών, και πιθανών σημείων κόπωσης ή εγκατάλειψης.

1) Ενσωμάτωση **Google Analytics** ή Mixpanel για καταγραφή custom events: level_start, level_complete, hint_used, bonus_completed.

2) Ανάλυση **retention funnels** (ποσό χρηστών που προχωρούν πέρα από το επίπεδο 5, 10 κ.λπ.) και A/B testing νέων χαρακτηριστικών.

Με αυτές τις προσθήκες, η εφαρμογή αποκτά ισχυρό περιβάλλον παραμετροποίησης, καλύτερη προσβασιμότητα και δυνατότητα μελλοντικής κλιμάκωσης—τόσο τεχνικά (νέα modules, online λειτουργίες), όσο και σε επίπεδο εμπειρίας χρήστη.

8.3 Προσαρμογή επιπέδων δυσκολίας

Μπορεί να προσαρμοστεί η δυσκολία των επιπέδων τροποποιώντας τις παραμέτρους:

- Πλήθος γραμμάτων στο σετ (generateBalancedLetters(n))
- Ελάχιστο/μέγιστο μήκος λέξεων (minWordLength, maxWordLength)
- Πλήθος απαιτούμενων λέξεων στο grid

8.4 Εμφανισιακές τροποποιήσεις (UI)

Το UI μπορεί να παραμετροποιηθεί μέσω των αρχείων του Pixi.js, αλλά και μέσω CSS/HTML wrapper. Μπορούν να αλλάξουν:

- Χρώματα φόντου, κουμπιών και γραμματοσειρών
- Τοποθέτηση και μέγεθος στοιχείων
- Προσθήκη ή αλλαγή εικόνων background/overlay

8.5 Προσθήκη νέων σκηνών / λειτουργιών

Η επεκτασιμότητα της εφαρμογής επιτρέπει την εύκολη προσθήκη:

- Νέων bonus τύπων (π.χ. scramble λέξεις, quiz)
- Πολλαπλών λεξικών ή εναλλαγής γλωσσών
- User accounts για παρακολούθηση προόδου

Οι νέες σκηνές μπορούν να υλοποιηθούν ως νέα JS αρχεία και να ενσωματωθούν μέσω του game.js.

8.6 Προσαρμογή για εκπαιδευτικά περιβάλλοντα

Το παιχνίδι μπορεί να προσαρμοστεί ώστε να ενσωματωθεί σε:

- Σχολικά εργαστήρια πληροφορικής
- Διαδραστικούς πίνακες (smartboards)
- Online μαθήματα μέσω εκπαιδευτικών portals

Προσθήκες όπως αξιολόγηση, έλεγχος χρόνου και καταγραφή αποτελεσμάτων μπορούν να υλοποιηθούν με απλά modules.

Η ευελιξία της παραμετροποίησης καθιστά το *το* σταυρόλεξο ιδιαίτερα κατάλληλο για μελλοντικές επεκτάσεις, προσαρμογές και χρήση σε ποικιλία σεναρίων.

8.7 Διεθνοποίηση και Πολυγλωσσική Υποστήριξη

Η εφαρμογή μπορεί να επεκταθεί ώστε να υποστηρίζει και άλλες γλώσσες, πέρα από την ελληνική. Για την υλοποίηση απαιτούνται τα εξής:

- Δημιουργία αντίστοιχων λεξικών JSON (π.χ. dictionary_en.json, dictionary_fr.json)
- Δυνατότητα επιλογής γλώσσας από τον χρήστη στην αρχή του παιχνιδιού (dropdown menu ή settings)
- Φόρτωση του κατάλληλου λεξικού κατά την εκκίνηση

Η λειτουργία αυτή ενισχύει τη διαπολιτισμική χρήση του παιχνιδιού και δίνει τη δυνατότητα εκπαιδευτικής αξιοποίησης σε πολυγλωσσικά περιβάλλοντα.

8.8 Responsive Σχεδίαση και Υποστήριξη Mobile

Η εφαρμογή μπορεί να βελτιστοποιηθεί περαιτέρω για χρήση σε κινητά και tablets:

- Αυτόματη αλλαγή διατάξεων ανάλογα με την ανάλυση της οθόνης
- Μεγαλύτερα πλήκτρα και padding για touch interactions
- Χρήση window.devicePixelRatio για καθαρό rendering
- Υποστήριξη gestures (tap, swipe) για εντοπισμό γραμμάτων

Η προσαρμογή των σκηνών LevelScene και LetterPalete με media queries ή προσαρμοστικές μεταβλητές διασφαλίζει άνετη εμπειρία χρήσης σε οποιαδήποτε συσκευή.

8.9 Αλγόριθμος Δημιουργίας Πλέγματος με Pangram

Μία από τις βασικές βελτιώσεις που ενσωματώθηκαν στη δημιουργία των επιπέδων αφορά τη χρήση μιας λέξης-σπόρου (ή αλλιώς pangram) για τη διασφάλιση πληρότητας και συνοχής της παλέτας γραμμάτων και του σταυρόλεξου.

Η λογική του προηγμένου αλγορίθμου περιλαμβάνει τα εξής βήματα:

1. Επιλογή Λέξης-Σπόρου

Η διαδικασία δημιουργίας ενός επιπέδου ξεκινά με την αναζήτηση μιας λέξης στο λεξικό που πληροί τα εξής κριτήρια:

- Έχει ακριβώς τον αριθμό γραμμάτων που απαιτείται για το επίπεδο (numLettersInPalette)
- Όλα τα γράμματά της είναι μοναδικά (χωρίς επαναλήψεις)

Αυτό εξασφαλίζει ότι η παλέτα θα είναι πλήρης και ισορροπημένη. Αν δεν βρεθεί άμεσα κατάλληλη λέξη, η αναζήτηση επαναλαμβάνεται για προκαθορισμένο αριθμό προσπαθειών.

2. Δημιουργία Παλέτας

Αφού βρεθεί η λέξη-σπόρος, γίνεται ανακάτεμα των γραμμάτων της και αυτά σχηματίζουν την παλέτα του επιπέδου, διασφαλίζοντας ότι όλα τα γράμματα της είναι χρήσιμα.

3. Εύρεση Υπόλοιπων Λέξεων

Με βάση αυτή την παλέτα, η εφαρμογή ψάχνει στο λεξικό για:

- Όλες τις λέξεις που μπορούν να σχηματιστούν με τα δεδομένα γράμματα
- Λέξεις εντός του εύρους δυσκολίας (μήκος, συχνότητα)

4. Έλεγχος Εφικτότητας

Ο αλγόριθμος ελέγχει αν:

- Ο αριθμός συνολικών λέξεων (μαζί με τη λέξη-σπόρο) είναι επαρκής για την κατασκευή ενός πλέγματος
- Οι λέξεις πληρούν τα ελάχιστα όρια μήκους και πληρότητας

Αν αποτύχει, η διαδικασία ξεκινά από την αρχή.

5. Δημιουργία Σταυρολέξου με mustIncludeWord

Η συνάρτηση generateCrossword() (στο αρχείο wordGenerator.js) τροποποιήθηκε ώστε να δέχεται προαιρετική παράμετρο mustIncludeWord, που επιβάλλει την τοποθέτηση της λέξης-σπόρου μέσα στο grid. Αυτό διασφαλίζει ότι:

- Η βασική λέξη του επιπέδου θα υπάρχει στο σταυρόλεξο
- Όλα τα γράμματα της παλέτας χρησιμοποιούνται ορθά

Αν το πλέγμα δεν μπορέσει να συμπεριλάβει τη λέξη-σπόρο, η διαδικασία δημιουργίας επαναλαμβάνεται εξ ολοκλήρου.

8.10 Ενσωμάτωση Τεχνητής Νοημοσύνης (AI)

Η εφαρμογή, λόγω της modular αρχιτεκτονικής της, μπορεί να επεκταθεί ώστε να αξιοποιεί τεχνικές Τεχνητής Νοημοσύνης (Artificial Intelligence) και Μηχανικής Μάθησης (Machine Learning) με στόχο την ενίσχυση της εμπειρίας χρήστη, την εξατομίκευση και την βελτιστοποίηση της δημιουργίας επιπέδων.

8.10.1 Έξυπνος Αλγόριθμος Δημιουργίας Επιπέδων

Μια από τις πιο προφανείς χρήσεις AI αφορά την **προσαρμογή των επιπέδων στον χρήστη**. Αντί για στατική ή γραμμικά αυξανόμενη δυσκολία, θα μπορούσε να εφαρμοστεί ένας αλγόριθμος που μαθαίνει από τη συμπεριφορά του παίκτη:

- Ανάλυση ιστορικών στοιχείων: χρόνος ολοκλήρωσης, χρήση βοήθειας, ποσοστό σωστών/λανθασμένων λέξεων.
- Δημιουργία **προφίλ παίκτη** (beginner, advanced, speed player κ.λπ.)
- Επιλογή λέξεων και μεγέθους επιπέδου με βάση το προφίλ

Μέθοδος: Χρήση supervised learning (π.χ. Random Forest ή μικρό νευρωνικό δίκτυο) σε server-side Python script που δημιουργεί δυναμικά dictionary και difficulty score

8.10.2 Natural Language Processing για Σημασιολογική Ανάλυση Λέξεων

Η λέξη "ζάρι" και η λέξη "κύβος" θεωρούνται διαφορετικές, αν και έχουν παρόμοιο νόημα. Η χρήση **NLP τεχνικών** θα μπορούσε να επιτρέψει:

- **Ομαδοποίηση λέξεων** με παρόμοιο νόημα (semantic clusters)
- Αναγνώριση συνωνύμων και εμφάνιση εναλλακτικών λέξεων
- Παροχή λεκτικής βοήθειας στον παίκτη με σημασιολογική καθοδήγηση ("σκέψου λέξη σχετική με...")

Μέθοδος*:* Ενσωμάτωση μοντέλων όπως fastText ή BERT εκπαιδευμένα σε ελληνικά, για σημασιολογικές συσχετίσεις.

8.10.3 Αυτόματη Επέκταση Λεξικού με LLM (Large Language Models)

Χρήση AI (π.χ. GPT) για:

- **Αυτόματη πρόταση λέξεων** για εμπλουτισμό του λεξικού (π.χ. σπάνιες λέξεις, ορολογία θεματικών ενοτήτων)
- Εντοπισμός ασάφειας ή λαθών (π.χ. πολυσήμαντες λέξεις ή λανθασμένες ορθογραφίες)
- Κατηγοριοποίηση λέξεων ανά θεματικό πεδίο

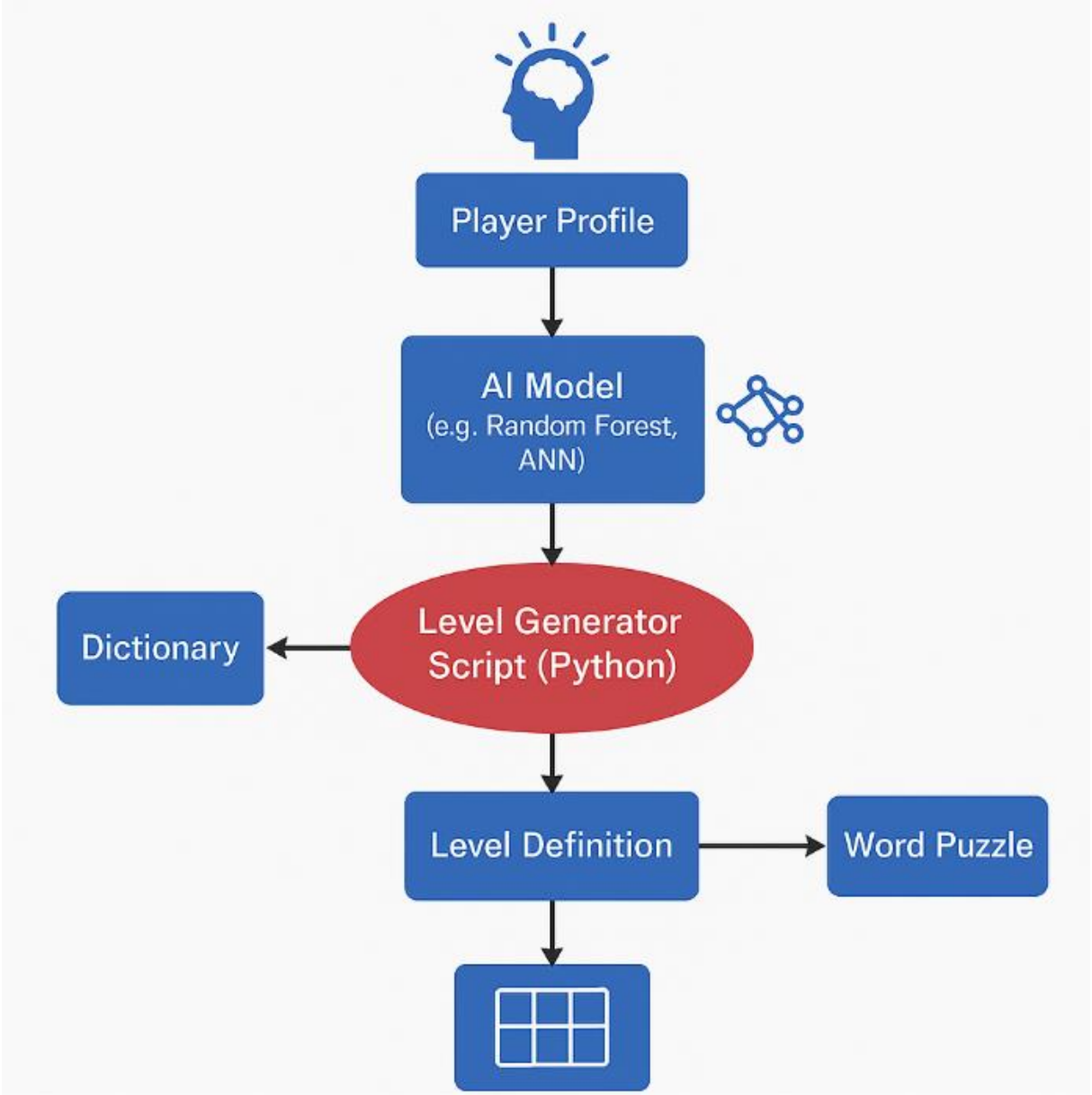
Μέθοδος: API με χρήση LLM (π.χ. OpenAI API) για εξαγωγή, φίλτρο και tagging λεξιλογίου.

8.10.4 Chatbot για Βοήθεια Παίκτη

Μια ακόμη δυνατότητα είναι η ενσωμάτωση ενός απλού **chatbot** που θα υποστηρίζει τον παίκτη σε πραγματικό χρόνο:

- Παροχή βοήθειας με φυσική γλώσσα (π.χ. "δείξε μου μια λέξη με τέσσερα γράμματα")
- Καθοδήγηση σε κανόνες και στρατηγικές
- Επεξήγηση άγνωστων λέξεων

Μέθοδος: Ενσωμάτωση web-based AI agent σε overlay UI. Μπορεί να αξιοποιηθεί κάποιο pretrained μοντέλο ή υπηρεσία chatbot (π.χ. DialogFlow, Rasa ή OpenAI Assistant API).



Εικόνα 63: Διάγραμμα ροής για AI-βασισμένη υποστήριξη λέξεων με ενσωμάτωση GPT

Οφέλη από την Ενσωμάτωση AI

1) Εξατομίκευση εμπειρίας χρήστη

2) Εκπαιδευτική αξιοποίηση (λέξεις-στόχοι, γνωστικό feedback)

3) Προσαρμογή δυσκολίας σε πραγματικό χρόνο

4) Εμπλουτισμός λεξιλογίου και θεματική ταξινόμηση

Κεφάλαιο 9ο: Συμπεράσματα

Η παρούσα εργασία παρουσίασε την ανάπτυξη μιας πλήρως δυναμικής εφαρμογής σταυρολέξου, σχεδιασμένης εξ ολοκλήρου με τεχνολογίες ιστού. Μέσα από τη συνδυαστική χρήση JavaScript, Pixi.js, GSAP για animations, JSON για διαχείριση λεξικών και Python για προεπεξεργασία λέξεων, δημιουργήθηκε ένα σύστημα που προσαρμόζεται αυτόματα σε κάθε νέα εκτέλεση του παιχνιδιού, προσφέροντας πάντα μοναδικές παλέτες γραμμάτων και επίπεδα διαφορετικής δυσκολίας.

Κύρια Επιτεύγματα

- **Δυναμική Δημιουργία Επίπεδων:** Η λογική επιλογής λέξης-σπόρου και η χρήση Python (βιβλιοθήκη wordfreq) για φιλτράρισμα και κανονικοποίηση ελληνικών λέξεων εξασφαλίζουν ποικιλία και ποιότητα στο παραγόμενο λεξικό.
- **Responsive & Interactive UI:** Με την Pixi.js και rendering σε καμβά, το πλέγμα, οι παλέτες γραμμάτων και τα βοηθήματα (hints, comet, διαμάντια, bonus επίπεδα) παρουσιάζονται ομαλά και διαδραστικά, ενώ το GSAP παρέχει ομαλή κι ευέλικτη διαχείριση animation.
- **Διαχείριση Κατάστασης & Ιστορικού:** Η απλή αλλά αποτελεσματική χρήση του localStorage διασφαλίζει ότι κάθε συνδυασμός γραμμάτων εμφανίζεται μόνο μία φορά ανά run, ενώ επιτρέπει επαναφορά ή τροποποίηση σε πραγματικό χρόνο.
- **Εργαλεία Ανάπτυξης & Debugging:** Η ενσωμάτωση hooks όπως window.currentLevelScene και window.levelGen μαζί με τα DevTools του browser δίνει τη δυνατότητα για Live Inspection — από logs δημιουργίας επιπέδων μέχρι επεξεργασία localStorage — επιταχύνοντας τον εντοπισμό σφαλμάτων και τον πειραματισμό.

Οφέλη από τη Χρήση DevTools

Η hands-on προσέγγιση με τα Developer Tools απέδειξε στην πράξη:

- **Άμεση Επισκόπηση** των εσωτερικών δομών (λίστες λέξεων, πίνακας κελιών).

- **Ζωντανός Πειραματισμός** με παλέτες γραμμάτων (μέθοδος debugWithPalette) χωρίς αλλαγές στον κώδικα.
- **Έλεγχος & Τροποποίηση** του ιστορικού επιπέδων, του UI στυλ και του animation timing σε πραγματικό χρόνο.
- **Βελτίωση της Σχεδίασης** μέσω live CSS tweaks (Elements pane) και χρήση breakpoints / step-through (Sources pane) για κατανόηση της ροής.

Συνολική Εμπειρία & Προοπτικές Επέκτασης

Η ολοκλήρωση αυτού του έργου ανέδειξε τη σημασία:

- Της καθαρής αρχιτεκτονικής (modular components, separation of concerns).
- Της αποτελεσματικής διαχείρισης δεδομένων (λεξικά, ιστορικό, κατάσταση παιχνιδιού).
- Της ενσωμάτωσης εργαλείων ανάπτυξης από την αρχή του σχεδιασμού, ώστε το debugging και το tuning να είναι γρήγορα και αξιόπιστα.

Στο μέλλον, η εφαρμογή μπορεί να εμπλουτιστεί με:

- **Leaderboard & Analytics**, ώστε οι παίκτες να συγκρίνονται παγκόσμια.
- **Multiplayer Mode**, για συνεργατική ή ανταγωνιστική επίλυση σταυρολέξων.
- **Θεματικά Πακέτα Λεξικών**, π.χ. γεωγραφίας, επιστήμης, ιστορίας.
- **Πρόσθετα UI Features**, όπως dark mode, custom animations και accessibility options.

Με αυτόν τον τρόπο, η εφαρμογή όχι μόνο προσφέρει πλούσια εμπειρία παιχνιδιού, αλλά και ένα ευέλικτο περιβάλλον ανάπτυξης για μελλοντικές βελτιώσεις.

ΒΙΒΛΙΟΓΡΑΦΙΑ

Ο τρόπος γραφής των βιβλιογραφικών αναφορών γίνεται σύμφωνα με τα παρακάτω παραδείγματα (IEEE style):

[1] Mozilla Developer Network (MDN). JavaScript Reference. https://developer.mozilla.org/en-US/docs/Web/JavaScript

[2] PixiJS Official Documentation. https://pixijs.com/

[3] GSAP Animation Platform. https://gsap.com/

[4] W3Schools. HTML, CSS & JS Tutorials. https://www.w3schools.com/

[5] M. Prensky (2001). Digital Game-Based Learning. McGraw-Hill.

[6] Κ. Νικολαΐδης. Σχεδίαση Εκπαιδευτικών Παιχνιδιών με χρήση Web Τεχνολογιών. Πανεπιστήμιο Μακεδονίας, 2020.

[7] GitHub Repositories & Tutorials. Διαθέσιμο μέσω αναζήτησης στο https://github.com

[8] Stack Overflow – Κοινότητα προγραμματιστών για απορίες JavaScript και Canvas

[9] S. Bird, E. Klein, and E. Loper, Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit, O'Reilly Media, 2009.

[10] D. Jurafsky and J. H. Martin, Speech and Language Processing, 3rd ed., Stanford University, 2021. [Online]. Available: https://web.stanford.edu/~jurafsky/slp3/

[11] C. Kirov and R. Radulov, "Crossword puzzle generation using constraint satisfaction," International Journal of Advanced Computer Science and Applications (IJACSA), vol. 10, no. 7, pp. 400–405, 2019.

[12] Luminoso Insight, wordfreq – A frequency dictionary tool for many languages. GitHub Repository, 2022. [Online]. Available: https://github.com/LuminosoInsight/wordfreq

[13] E. Wasserman and R. Kleinberg, "Learning to generate crosswords using structure-based features," Proceedings of the AAAI Conference on Artificial Intelligence, 2011.

ΠΑΡΑΡΤΗΜΑ Α : Κώδικας Υλοποίησης BonusScene

Ο παρακάτω κώδικας παρουσιάζει τη βασική δομή του Bonus level

```
import { Container, Text, Graphics } from 'pixi.js';
import gsap from 'gsap';
import LetterPalete from '../Sections/letterPalete';
import { GAME_WIDTH, GAME_HEIGHT } from '..';

// Define costs/rewards
const DIAMOND_REWARD = 1;
const DIAMOND_PENALTY = 1;
const POINT_PENALTY = 10;
const BONUS_TIME = 30; // seconds

export default class BonusScene extends Container {
    constructor(letters, targetWord, gameInstance) {
        super();
        this.letters = letters;
        this.targetWord = targetWord;
        this.game = gameInstance; // Reference to main game for score/diamonds

        this.timer = BONUS_TIME;
        this.timeLeft = BONUS_TIME;
        this.wordFound = false;
        this.isComplete = false;

        // Callback for completion
        this.onComplete = null;

        // Graphics object for the timer arc
        this.timerArc = new Graphics();

        this.initUI();
```

```
        this.startTimer();
    }

    initUI() {
        console.log(`Bonus Scene: Find "${this.targetWord}" with [${this.letters.join(',')}]`);

        // Background (Slightly less intense purple)
        const background = new Graphics();
        background.beginFill(0x5A0CAD, 0.95); // Slightly adjusted purple
        background.drawRect(0, 0, GAME_WIDTH, GAME_HEIGHT);
        background.endFill();
        this.addChild(background);

        // Title Banner (More styled - Placeholder graphics)
        const bannerHeight = 60;
        const bannerY = 30;
        const titleBanner = new Graphics();
        titleBanner.beginFill(0x9400D3, 1);
        // Simple ribbon-like shape approximation
        titleBanner.moveTo(0, bannerY);
        titleBanner.lineTo(GAME_WIDTH, bannerY);
        titleBanner.lineTo(GAME_WIDTH - 20, bannerY + bannerHeight / 2);
        titleBanner.lineTo(GAME_WIDTH, bannerY + bannerHeight);
        titleBanner.lineTo(0, bannerY + bannerHeight);
        titleBanner.lineTo(20, bannerY + bannerHeight / 2);
        titleBanner.closePath();
        titleBanner.endFill();
        this.addChild(titleBanner);

        const titleText = new Text("ΠΡΟΚΛΗΣΗ ΜΙΑΣ ΛΕΞΗΣ", {
            fill: 0xFFFFFF, fontSize: 28, fontWeight: 'bold',
            stroke: 0x4B0082, strokeThickness: 4 // Darker purple stroke
        });
        titleText.anchor.set(0.5);
        titleText.x = GAME_WIDTH / 2;
        titleText.y = bannerY + bannerHeight / 2;
        this.addChild(titleText);

        // Diamond Display Area (Top Right)
        const diamondAreaX = GAME_WIDTH - 110;
        const diamondAreaY = bannerY + bannerHeight + 20;
        const diamondBg = new Graphics();
        diamondBg.beginFill(0x000000, 0.4);
        diamondBg.drawRoundedRect(diamondAreaX - 10, diamondAreaY - 5, 100, 40, 10);
        diamondBg.endFill();
        this.addChild(diamondBg);
        // Use Unicode escape for diamond emoji
        this.diamondText = new Text(`${this.game.totalDiamonds}\u{1F48E}`, { fill: 0xFFFFFF,
fontSize: 20, fontWeight: 'bold' });
        this.diamondText.anchor.set(1, 0.5);
        this.diamondText.x = diamondAreaX + 85;
        this.diamondText.y = diamondAreaY + 15;
        this.addChild(this.diamondText);

        // Timer Display Area (Top Center)
```

```
const timerAreaY = bannerY + bannerHeight + 70;
const timerRadius = 45; // Slightly smaller radius

// Timer Display Area (Top Center)
const timerBg = new Graphics();
timerBg.beginFill(0x000000, 0.4);
timerBg.drawCircle(GAME_WIDTH / 2, timerAreaY + 15, timerRadius);
timerBg.endFill();
this.addChild(timerBg);

// Add the arc graphic object
this.timerArc.x = GAME_WIDTH / 2;
this.timerArc.y = timerAreaY + 15;
this.addChild(this.timerArc);
this.#drawTimerArc(1); // Draw initial full arc

this.timerText = new Text(`${this.timeLeft}s`, {
    fill: 0xFFFFFF, fontSize: 24, fontWeight: 'bold'
});
this.timerText.anchor.set(0.5);
this.timerText.x = GAME_WIDTH / 2;
this.timerText.y = timerAreaY + 15;
this.addChild(this.timerText);

// --- Target Word Underscores --- //
const underscoreContainer = new Container();
const underscoreSpacing = 15;
const underscoreStyle = { fill: 0xFFFFFF, fontSize: 50, fontWeight: 'bold' };
let totalWidth = 0;

for (let i = 0; i < this.targetWord.length; i++) {
    const underscoreText = new Text('_', underscoreStyle);
    underscoreText.x = totalWidth; // Position based on accumulated width
    underscoreContainer.addChild(underscoreText);
    // Add width of underscore plus spacing for the next one
    totalWidth += underscoreText.width + underscoreSpacing;
}
// Remove the last spacing
totalWidth -= underscoreSpacing;

// Set the calculated width to the container for centering
underscoreContainer.width = totalWidth;

// Center the underscores container
underscoreContainer.x = (GAME_WIDTH - underscoreContainer.width) / 2;
underscoreContainer.y = timerAreaY + 90; // Position below timer
this.addChild(underscoreContainer);
// ------------------------------- //

// Remove debug text showing the target word
// const targetDisplay = new Text(`(Βρες: ${this.targetWord})`, { fill: 0xFFFFFF, fontSize: 20 });
// targetDisplay.anchor.set(0.5);
// targetDisplay.x = GAME_WIDTH / 2;
// targetDisplay.y = 200;
// this.addChild(targetDisplay);
```

```
    // Letter Palette (Positioned lower-right)
    this.letterPalet = new LetterPalete(this.letters, this.#bonusWordCheck.bind(this));
    const paletteX = GAME_WIDTH - this.letterPalet.width - 40; // 40px padding from right edge
    const paletteY = GAME_HEIGHT - this.letterPalet.height - 40; // 40px padding from bottom edge
    this.letterPalet.position.set(paletteX, paletteY);
    this.addChild(this.letterPalet);

    // TODO: Add proper Start Button graphic/text
    // TODO: Add clock graphic to banner?
    // TODO: Add diamond sprites?
}

// Callback for the LetterPalette
#bonusWordCheck(letterPalet) {
    const submittedWord = letterPalet.lastSelectionText;
    if (!submittedWord || this.isComplete) {
        letterPalet.lastSelectionText = "";
        return;
    }

    if (submittedWord === this.targetWord) {
        console.log("[BonusScene] Correct word submitted! Calling #completeBonus(true)");
        this.wordFound = true;
        this.#completeBonus(true); // Success
    }

    letterPalet.lastSelectionText = "";
}

startTimer() {
    this.ticker = gsap.ticker.add(this.#updateTimer.bind(this));
}

#updateTimer(time, deltaTime) {
    if (this.isComplete) return;

    this.timeLeft -= deltaTime / 1000;
    const displayTime = Math.max(0, Math.ceil(this.timeLeft));
    this.timerText.text = `${displayTime}s`;

    // Update the timer arc
    const progress = Math.max(0, this.timeLeft / BONUS_TIME);
    this.#drawTimerArc(progress);

    if (this.timeLeft <= 0) {
        this.timeLeft = 0;
        this.timerText.text = "0s";
        console.log("Bonus Time Up!");
        this.#completeBonus(false); // Failure
    }
}

// Helper to draw the timer arc
#drawTimerArc(progress) { // progress is 0.0 to 1.0
```

```
        const radius = 45;
        const startAngle = -Math.PI / 2; // Start from the top
        const endAngle = startAngle + (progress * Math.PI * 2);

        this.timerArc.clear();
        this.timerArc.lineStyle(5, 0xFFFFFF, 1); // White line for the arc
        this.timerArc.arc(0, 0, radius - 2.5, startAngle, endAngle, false); // Draw arc
    }

    #completeBonus(success) {
        console.log(`[BonusScene] #completeBonus entered with success: ${success}`);
        if (this.isComplete) return;
        this.isComplete = true;
        gsap.ticker.remove(this.ticker);
        this.letterPalet.interactiveChildren = false;

        let message = "";
        let details = {};

        // --- DEBUG LOG ---
        console.log("[BonusScene] Checking this.game before update:", this.game);
        if (this.game) {
            console.log("[BonusScene] Type of this.game.updateDiamonds:", typeof
this.game.updateDiamonds);
        } else {
            console.error("[BonusScene] ERROR: this.game is not defined here!");
        }
        // -----------------
```

ΠΑΡΑΡΤΗΜΑ Β : Κώδικας Υλοποίησης letterPallete

Η παρακάτω κλάση αναπαριστά το στήσιμο της παλέτας γραμμάτων.

```
import { Container, Graphics, Sprite } from "pixi.js";
import { COLORS, GAME_CELLSIZE } from "..";
import PalletCell from "../GameObjects/palletCell";
import Shuffle from "../GameObjects/shuffleButton";
import gsap from "gsap";

export default class LetterPalete extends Container {
  // Ο constructor δέχεται τώρα levelLetters και το callback
  constructor(levelLetters, gridControlCallBack) {
    super();

    this.levelLetters = levelLetters;  // Αποθηκεύουμε τα γράμματα του επιπέδου
    this.gridControlCallBack = gridControlCallBack;

    this.bubbleSprite = Sprite.from("bubble");
    this.paleteCells = [];
    this.selectionList = [];
    this.lines = [];
```

```
  this.palletLettersPositions = [];
  this.selectionStart = false;
  this.lastSelectionText = "";

  this.shuffleButton = new Shuffle();
  this.shuffleButton.zIndex = 1;
  this.shuffleButton.interactive = true;
  this.shuffleButton.on('pointerdown', () => {
    this.shuffleLetters();
  });
  this.addChild(this.shuffleButton);

  this.selectedLetterDisplayContainer = new Container();
  this.init();
}

init() {
  this.sortableChildren = true;
  this.#initBg();
  this.#initLetters();
  this.addChild(this.selectedLetterDisplayContainer);
}

// Υπολογισμός του κατάλληλου μεγέθους του τροχού με βάση τον αριθμό των γραμμάτων
#calculateOptimalSize() {
  const numLetters = this.levelLetters.length;

  // Βασικό μέγεθος για τον τροχό
  let baseSize = GAME_CELLSIZE * 3.2;

  // Προσαρμογή μεγέθους ανάλογα με τον αριθμό γραμμάτων
  if (numLetters <= 5) {
    // Για λίγα γράμματα, μικρότερος τροχός
    return baseSize * 0.8;
  } else if (numLetters <= 8) {
    // Κανονικό μέγεθος για 6-8 γράμματα
    return baseSize;
  } else if (numLetters <= 10) {
    // Λίγο μεγαλύτερος τροχός για 9-10 γράμματα
    return baseSize * 1.2;
  } else {
    // Ακόμα μεγαλύτερος τροχός για 11+ γράμματα
    // Η αύξηση είναι 10% για κάθε επιπλέον 2 γράμματα πάνω από τα 10
    const extraSize = Math.ceil((numLetters - 10) / 2) * 0.1;
    return baseSize * (1.2 + extraSize);
  }
}

#initBg() {
  // Υπολογίζουμε το κατάλληλο μέγεθος για τον τροχό
  const bubbleSize = this.#calculateOptimalSize();

  this.bubbleSprite.width = bubbleSize;
  this.bubbleSprite.height = bubbleSize;
  this.bubbleSprite.anchor.set(0.5);
```

```
  this.bubbleSprite.alpha = 1.0;
  this.bubbleSprite.zIndex = 0;
  this.addChild(this.bubbleSprite);
}

#initLetters() {
  const numLetters = this.levelLetters.length;

  // Υπολογίζουμε την ακτίνα με βάση το μέγεθος του bubble και τον αριθμό των γραμμάτων
  const bubbleSize = this.bubbleSprite.width;

  // Προσαρμόζουμε την ακτίνα ώστε τα γράμματα να μην είναι πολύ κοντά σε μεγάλο αριθμό
  // αλλά και να μην είναι πολύ μακριά σε μικρό αριθμό
  let radiusMultiplier = 0.35; // Βασικός πολλαπλασιαστής

  if (numLetters <= 5) {
    radiusMultiplier = 0.3; // Πιο κοντά στο κέντρο για λίγα γράμματα
  } else if (numLetters > 10) {
    // Αυξάνουμε σταδιακά για περισσότερα γράμματα για να μην είναι πολύ στριμωγμένα
    radiusMultiplier = 0.35 + (numLetters - 10) * 0.01;
  }

  const radius = bubbleSize * radiusMultiplier;
  const angleIncrement = (Math.PI * 2) / numLetters;

  for (let i = 0; i < numLetters; i++) {
    const angle = angleIncrement * i;
    const x = radius * Math.cos(angle);
    const y = radius * Math.sin(angle);

    this.palletLettersPositions.push({ x, y });

    const palletCell = new PalletCell(
      this.levelLetters[i],
      this.#onClickPalletCell.bind(this),
      this.#onMovePalletCell.bind(this),
      this.#onPointerUp.bind(this)
    );
    palletCell.zIndex = 3;
    palletCell.position.set(x, y);
    this.paleteCells.push(palletCell);
    this.addChild(palletCell);
  }
}

#resetLetterPallete() {
  this.removeLines();
  this.selectionList = [];
  this.selectionStart = false;
  this.#resetPalletCells();
  this.draw = false;
}

#resetPalletCells() {
  this.paleteCells.forEach(cell => cell.reset());
```

```
}

#onPointerUp() {
  // Φτιάχνουμε το lastSelectionText από τα κελιά που επιλέχθηκαν
  this.selectionList.forEach(sel => {
    this.lastSelectionText += sel.letter.letter;
  });
  // Καλούμε το callback στο GridSection
  if (this.gridControlCallBack) {
    this.gridControlCallBack(this);
  }
  this.#resetLetterPallete();
}

#onMovePalletCell(hoverCell) {
  if (this.selectionStart && !hoverCell.unlock) {
    this.selectionList.push(hoverCell);
    hoverCell.unlock = true;
  }
}

#onClickPalletCell(clickedCell) {
  if (!clickedCell.unlock) {
    this.selectionList.push(clickedCell);
    clickedCell.unlock = true;
    this.selectionStart = true;
  }
}

removeLines() {
  if (!this.draw && this.lines.length > 0) {
    this.lines.forEach(line => {
      line.visible = false;
    });
  }
}

drawLine() {
  if (this.selectionList.length > 0) {
    this.draw = true;
    for (let i = this.selectionList.length - 1; i > 0; i--) {
      const last = this.selectionList[i];
      const lastPrev = this.selectionList[i - 1];
      if (lastPrev && (!last.drawed || !lastPrev.drawed)) {
        let line = new Graphics();
        line.lineStyle(15, COLORS.orange);
        line.moveTo(last.x, last.y);
        line.lineTo(lastPrev.x, lastPrev.y);
        line.zIndex = 2;
        this.addChild(line);
        this.lines.push(line);
        last.drawed = true;
        lastPrev.drawed = true;
      }
    }
```

```
    }
  }

  shuffleLetters() {
    if (this.shuffleButton.shuffle) {
      let tempPos = this.#shuffleArray(this.palletLettersPositions);
      for (let i = 0; i < tempPos.length; i++) {
        this.#animateLetterMovement(this.paleteCells[i], tempPos[i]);
      }
      this.shuffleButton.shuffle = false;
    }
  }

  #shuffleArray(array) {
    for (let i = array.length - 1; i > 0; i--) {
      const j = Math.floor(Math.random() * (i + 1));
      [array[i], array[j]] = [array[j], array[i]];
    }
    return array;
  }

  #animateLetterMovement(letter, newPosition) {
    gsap.to(letter, { duration: 0.6, x: newPosition.x, y: newPosition.y });
  }

  update(delta) {
    this.removeLines();
    this.drawLine();
    this.paleteCells.forEach(cell => cell.update());
    if (this.selectionStart) {
      this.shuffleButton.visible = false;
    } else {
      this.shuffleButton.visible = true;
    }
  }
}
```

ΠΑΡΑΡΤΗΜΑ C : Κώδικας Υλοποίησης get_words.py

Ακολουθεί ο πλήρης κώδικας του αρχείου get_words.py

```
import sys
import unicodedata # Import unicodedata for character properties

try:
    from wordfreq import top_n_list

    print("Fetching top 50,000 Greek words from wordfreq...")
    # You can adjust the number 50000 if needed
    greek_words_freq = top_n_list('el', 50000)
    print(f"Fetched {len(greek_words_freq)} words.")
```

```
    def normalize_greek_for_dictionary(text):
        if not text: return ""
        # Remove diacritics (tones)
        nfkd_form = unicodedata.normalize('NFD', text)
        # Keep only letters, convert to uppercase
        return "".join([c for c in nfkd_form if unicodedata.category(c) == 'Lu' or unicodedata.category(c)
== 'Ll']).upper()

    filtered_words = set()
    processed_count = 0
    for word in greek_words_freq:
        processed_count += 1
        # Normalize first (remove accents, keep only letters, then uppercase)
        normalized_word = normalize_greek_for_dictionary(word)

        if 3 <= len(normalized_word) <= 12 and normalized_word.isalpha(): # isalpha check after
normalization
            filtered_words.add(normalized_word)
        # else:
            # if processed_count % 5000 == 0: # Log occasionally if needed
            #     print(f"Skipped or empty after normalization: '{word}' -> '{normalized_word}'")

    print(f"Processed {processed_count} words from wordfreq list.")

    sorted_words = sorted(list(filtered_words))

    # --- Output to a file ---
    output_filename = "new_dictionary_content.txt" # Changed name for clarity
    word_count = len(sorted_words)
    print(f"\nTotal words for dictionary (3-12 letters, Greek only, no accents): {word_count}")
    print(f"Saving words to {output_filename} in full dictionary.json format...")

    with open(output_filename, "w", encoding="utf-8") as f:
        f.write("{\n")
        f.write("  \"words\": [\n") # Start the JSON structure
        for i, word_to_write in enumerate(sorted_words):
            # Ensure proper escaping for JSON strings
            escaped_word = word_to_write.replace('"', '\\"')
            # Write word with comma, except for the last one
            f.write(f'    "{escaped_word}"{"," if i < word_count - 1 else ""}\n')
        f.write("  ]\n") # Close the words array
        f.write("}\n") # Close the main JSON object

    print(f"Successfully saved {word_count} words to {output_filename}.")
    print(f"You can now rename {output_filename} to dictionary.json or copy its content to replace the
existing dictionary.json.")

except ImportError:
    print("\n--- ERROR ---")
    print("The 'wordfreq' library is not installed.")
    print("Please install it first:")
    print("pip install wordfreq")
    print("-------------")
except Exception as e:
```

```
    print(f"An error occurred: {e}")
```

ΠΑΡΑΡΤΗΜΑ D : Κώδικας Υλοποίησης assets.js

Ακολουθεί ο πλήρης κώδικας του αρχείου assets.js που αρχικοποιεί τους πόρους του παιχνιδιού, την παρακολούθηση της κατάστασης φόρτωσης και την διαχείριση bundles μεσω της βιβλιοθήκης Pixi.js.

```
import {
  Assets,
  extensions,
  ExtensionType,
  resolveTextureUrl,
  settings,
} from "pixi.js";

import manifest from "./manifest.json";

export const resolveJsonUrl = {
  extension: ExtensionType.ResolveParser,
  test: (value) =>
    settings.RETINA_PREFIX.test(value) && value.endsWith(".json"),
  parse: resolveTextureUrl.parse,
};

extensions.add(resolveJsonUrl);

export async function initAssets() {
  await Assets.init({ manifest });
  const bundleList = ["game", "fonts"];

  const assets = await Assets.loadBundle(bundleList);

  const gameBundles = manifest.bundles.map((item) => item.name);

  Assets.backgroundLoadBundle(gameBundles);

  return assets.default ? assets.default : assets;
}

export function isBundleLoaded(bundle) {
  const bundleManifest = manifest.bundles.find((b) => b.name === bundle);

  if (!bundleManifest) {
    return false;
  }

  for (const asset of bundleManifest.assets) {
    if (!Assets.cache.has(asset.name)) {
      return false;
    }
  }
```

```
  return true;
}

export function areBundlesLoaded(bundles) {
  for (const name of bundles) {
    if (!isBundleLoaded(name)) {
      return false;
    }
  }

  return true;
}
```

ΠΑΡΑΡΤΗΜΑ Ε : Εργαλεία και Βιβλιοθήκες

Κατά την ανάπτυξη της εφαρμογής αξιοποιήθηκαν τα παρακάτω εργαλεία, βιβλιοθήκες και τεχνολογίες:

- **JavaScript (ES6+)**: Βασική γλώσσα ανάπτυξης λογικής και αρχιτεκτονικής
- **Pixi.js**: Για το 2D rendering της διεπαφής (πλέγμα, κουμπιά, animations)
- **GSAP (GreenSock Animation Platform)**: Για όλα τα animation (transitions, fades, scaling)
- **HTML5 / CSS3**: Για την ενσωμάτωση του καμβά και βασικές ρυθμίσεις εμφάνισης
- **JSON**: Για τη διαχείριση δεδομένων (λεξικό, στατικά επίπεδα, δομή)
- **Python 3.x**: Για τη δημιουργία καθαρού λεξικού μέσω του script `get_words.py`
- **VSCode**: Για επεξεργασία κώδικα και χρήση local server (Live Server extension)
- **Web browser (Chrome)**: Για την εκτέλεση και δοκιμή της εφαρμογής

Επιπλέον χρησιμοποιήθηκαν βοηθητικές βιβλιοθήκες όπως:

- wordfreq (Python): Για ανάκτηση συχνών ελληνικών λέξεων
- unicodedata (Python): Για καθαρισμό και κανονικοποίηση χαρακτήρων
- localStorage API: Για προσωρινή αποθήκευση δεδομένων στο πρόγραμμα περιήγησης