ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ ΚΑΙ ΤΕΧΝΟΛΟΓΙΑΣ

ΜΕΤΑΠΤΥΧΙΑΚΗ ΕΞΕΙΔΙΚΕΥΣΗ ΣΤΑ ΠΛΗΡΟΦΟΡΙΑΚΑ ΣΥΣΤΗΜΑΤΑ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΑΝΑΠΤΥΞΗ ΠΕΡΙΒΑΛΛΟΝΤΟΣ ΥΠΟΛΟΓΙΣΤΙΚΗΣ ΣΚΕΨΗΣ ΜΕΣΩ ΤΗΣ ΕΚΜΑΘΗΣΗΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ

ΜΟΥΣΙΟΥ ΜΑΡΙΑ

ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ: ΤΑΜΠΟΥΡΗΣ ΕΥΘΥΜΙΟΣ

Θεσσαλονίκη, Σεπτέμβριος 2021

Η παρούσα Εργασία καθώς και τα αποτελέσματα αυτής, αποτελούν συνιδιοκτησία του ΕΑΠ και του φοιτητή, ο καθένας από τους οποίους έχει το δικαίωμα ανεξάρτητης χρήσης, αναπαραγωγής και αναδιανομής τους (στο σύνολο ή τμηματικά) για διδακτικούς και ερευνητικούς σκοπούς, σε κάθε περίπτωση αναφέροντας τον τίτλο και το συγγραφέα της Εργασίας καθώς και το όνομα του ΕΑΠ όπου εκπονήθηκε.

ΑΝΑΠΤΥΞΗ ΠΕΡΙΒΑΛΛΟΝΤΟΣ ΥΠΟΛΟΓΙΣΤΙΚΗΣ ΣΚΕΨΗΣ ΜΕΣΩ ΤΗΣ ΕΚΜΑΘΗΣΗΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ

ΜΟΥΣΙΟΥ ΜΑΡΙΑ

Επιτροπή Επίβλεψης Διπλωματικής Εργασίας

Επιβλέπων Καθηγητής:

Ταμπούρης Ευθύμιος

Καθηγητής

Τμήμα Εφαρμοσμένης Πληροφορικής

Πανεπιστήμιο Μακεδονίας

Συν-Επιβλέπων Καθηγητής:

Φωκά Αμαλία

Επίκουρη Καθηγήτρια

Τμήμα Εικαστικών Τεχνών & Επιστημών της Τέχνης

Πανεπιστήμιο Ιωαννίνων

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή μου κ. Ευθύμιο Ταμπούρη για την ευκαιρία που μου έδωσε να ασχοληθώ με ένα τόσο σύγχρονο και ενδιαφέρον αντικείμενο καθώς και για την καθοδήγηση του και την κατανόηση που έδειξε κατά την διάρκεια της εκπόνησης της διπλωματικής μου εργασίας.

Επίσης ευχαριστώ θερμά την υποψήφια διδάκτορα κ. Χριστίνα Τίκβα για την αμέριστη στήριξη της, την άμεση διαθεσιμότητά της και την καθοριστική συμβολή της στην ολοκλήρωση της εργασίας.

Τέλος, θα ήθελα να ευχαριστήσω τον σύζυγο μου Ιωάννη, για την στήριξη, την υπομονή και την ώθηση που μου έδινε σε κάθε δυσκολία που αντιμετώπιζα κατά την διάρκεια των μεταπτυχιακών μου σπουδών.

Περίληψη

Την τελευταία δεκαετία υπάρχει έντονο ενδιαφέρον για την ανάπτυξη δεξιοτήτων Υπολογιστικής Σκέψης μέσω της εκμάθησης προγραμματισμού. Πληθώρα εργαλείων έχουν αναπτυχθεί με στόχο την εκμάθηση προγραμματισμού σε μικρές ηλικίες όπως το Scratch, Kodu Game Lab, Blockly Games. Αρκετές έρευνες έχουν συμπεριλάβει τα παραπάνω εργαλεία καθώς και άλλα εκπαιδευτικά παιχνίδια για την δημιουργία οδηγιών διδασκαλίας και πλαισίων ανάπτυξης της Υπολογιστικής Σκέψης σε μαθητές. Ωστόσο δεν υπάρχει κάποιο εκπαιδευτικό εργαλείο που να υποστηρίζει την εκμάθηση και την αξιολόγηση ειδικά της Υπολογιστικής Σκέψης μέσω της εφαρμογής συγκεκριμένων παιδαγωγικών μεθόδων. Σκοπός της παρούσας διπλωματικής εργασίας είναι η σχεδίαση και η υλοποίηση ενός εκπαιδευτικού εργαλείου που θα ενσωματώνει έννοιες και πρακτικές της Υπολογιστικής Σκέψης και θα κάνει χρήση της Μάθησης με Υποστήριξη (Scaffolding) μέσω της υποστήριξης των μαθητών με κατάλληλη ανάδραση και συμβουλές. Το εργαλείο που αναπτύχθηκε ονομάζεται aMazeD και κάνει χρήση των τεχνολογιών Blockly, JavaScript, Python, τροποποιώντας και επεκτείνοντας τα παιχνίδια Blockly και εφαρμόζοντας τις διαστάσεις των Υπολογιστικών Εννοιών και Πρακτικών που συναντάμε στο πλαίσιο Υπολογιστικής Σκέψης των Brennan και Resnick. Για την ανάπτυξη του aMazeD μελετήθηκαν οι έννοιες, τα εργαλεία και τα πλαίσια ανάπτυξης της Υπολογιστικής Σκέψης καθώς και εκπαιδευτικά παιχνίδια που κάνουν χρήση των τεχνολογιών που αναφέρθηκαν παραπάνω. Στην εργασία περιγράφεται αρχικά η δομή του παιχνιδιού, ο τρόπος που είναι σχεδιασμένα τα επίπεδα και η βοήθεια που παρέχεται στον χρήστη ακολουθούμενη από μια αναλυτική παρουσίαση του κώδικα που αναπτύχθηκε ή τροποποιήθηκε για κάθε λειτουργία που συναντάμε στο aMazeD.

Λέξεις-Κλειδιά

Υπολογιστική Σκέψη, Blockly, JavaScript, Python, Μάθηση με Υποστήριξη, Σχεδιασμός και Διαχείριση Λογισμικού, Εκπαιδευτικά Παιχνίδια

Abstract

In the last decade there is an ongoing interest about the development of Computational Thinking (CT) skills through learning programming. A variety of tools have been developed for learning programming at an early age such as Scratch, Kodu Game Lab, Blockly Games. Several researches have included the above tools as well as other educational games in order to design teaching instructions and frameworks for the development of Computational Thinking in students. However, there is no educational tool that supports learning and in particular concepts and practices of Computational Thinking through the application of specific pedagogical methods using scaffolding features. The purpose of this dissertation is to design and implement an educational game for the development of Computational Thinking that will make use of Scaffolding through the support of students with appropriate feedback and advice. The tool that has been created is called aMazeD and uses Blockly, JavaScript and Python technologies. Moreover, the game is based on Blockly Games' Maze and Turtle and the logic of the game incorporates the dimensions of Computational Concepts and Practices found in Brennan and Resnick's framework. For the development of aMazeD, the concepts, tools and frameworks of Computational Thinking were studied as well as educational games that make use of the technologies mentioned above. In this work we initially present the game as a whole, its features and functionalities and the scaffoldings of the game, followed by a deep analysis of the code that was developed or modified in order to implement the desired standards of the game.

Keywords

Computational Thinking, Blockly, JavaScript, Python, Scaffolding, Software Design and Management, Educational Games

Περιεχόμενα

Κατάλογος Εικόνων / Σχημάτων / Κώδικα

Κατάλογος Πινάκων

Συντομογραφίες & Ακρωνύμια

Ακολουθούν κάποια παραδείγματα:

ΔΕ	Διπλωματική Εργασία
ΕΑΠ	Ελληνικό Ανοικτό Πανεπιστήμιο
ΘΕ	Θεματική Ενότητα
ΠΕ	Πτυχιακή Εργασία
ΠΣ	Πρόγραμμα Σπουδών
ΣΥΝ	Συντονιστής
ΥΣ	Υπολογιστική Σκέψη

1. Εισαγωγή

1.1. Περιγραφή του προβλήματος

Τα τελευταία χρόνια υπάρχουν πολλές αλλαγές στον τρόπο με τον οποίο διδάσκονται οι έννοιες της Πληροφορικής σε μικρές ηλικίες καθώς και στον σκοπό με τον οποίο χρησιμοποιείται η διδασκαλία του προγραμματισμού στην εκπαίδευση. Η εξέλιξη της τεχνολογίας, η εισαγωγή της ρομποτικής σε μικρές ηλικίες, δημοφιλή κινήματα όπως η Ώρα του Κώδικα έχουν φέρει στο προσκήνιο την ανάγκη για την ανάπτυξη όχι μόνο τεχνολογικών αλλά και πιο ουσιαστικών δεξιοτήτων όπως είναι η επίλυση προβλημάτων, η αξιολόγηση της λύσης ενός προβλήματος, η εύρεση μιας πιο αποδοτικής λύσης, ικανότητες που σχετίζονται άμεσα με την ανάπτυξη της Υπολογιστικής Σκέψης. Σε αυτόν τον τομέα έχουν αναπτυχθεί πολλά εκπαιδευτικά εργαλεία όπως τα Scratch, Kodu Game Lab, Alice για την εκμάθηση προγραμματισμού μέσω δημιουργίας παιχνιδιών και κινούμενων ιστοριών καθώς και αρκετά εκπαιδευτικά παιχνίδια, όπως τα Blockly Games, Run Marco!, CodeMonkey που εισάγουν τις έννοιες του προγραμματισμού σε αρχάριους μέσω εργαλείων οπτικού προγραμματισμού όπως είναι η βιβλιοθήκη Blockly. Τα περισσότερα από αυτά τα εργαλεία έχουν χρησιμοποιηθεί για την δημιουργία πλαισίων ανάπτυξης υπολογιστικής σκέψης αλλά δεν υπάρχει κάποιο εργαλείο που να έχει σχεδιαστεί εξ αρχής για να υποστηρίζει την εκμάθηση της υπολογιστικής σκέψης μέσω της εφαρμογής συγκεκριμένων παιδαγωγικών μεθόδων και με την χρήση συγκεκριμένων πρακτικών της μάθησης με υποστήριξη.

1.2. Αντικείμενο και στόχοι μελέτης

Η παρούσα εργασία πραγματεύεται την ανάπτυξη δεξιοτήτων ΥΣ μέσω της εκμάθησης προγραμματισμού. Εκτός αντικειμένου αποτελούν η αξιολόγηση της υπολογιστικής σκέψης και του εργαλείου που θα αναπτυχθεί ωστόσο για την πληρότητα της παρουσίασης του εργαλείου συμπεριλαμβάνεται η αξιολόγησή του στα πλαίσια άλλης έρευνας.

Ο βασικός στόχος της εργασίας είναι ο σχεδιασμός ενός περιβάλλοντος ΥΣ που θα ενσωματώνει την αξιολόγηση της ΥΣ μέσω της εφαρμογής συγκεκριμένων παιδαγωγικών μεθόδων, συμπεριλαμβάνοντας την Μάθηση με Υποστήριξη με την παροχή κατάλληλης ανάδρασης και συμβουλών στο χρήστη.

Επιμέρους στόχους αποτελούν:

- Η μελέτη και κατανόηση των βασικών εννοιών και πλαισίων αξιολόγησης της ΥΣ.
- Η μελέτη και κατανόηση των σχετικών τεχνολογιών ανάπτυξης Blockly, JavaScript, Python και της εφαρμογής τους στα παιχνίδια Blockly Games.
- Η μελέτη σχετικών εργαλείων ανάπτυξης ΥΣ και των παιδαγωγικών μεθόδων που εφαρμόζονται σε αυτά.

Η μελέτη της παρούσας εργασίας περιορίζεται σε περιβάλλοντα ανάπτυξης της ΥΣ που κάνουν χρήση εργαλείων οπτικού προγραμματισμού.

1.3. Περιεχόμενα μελέτης

Το δεύτερο κεφάλαιο της διπλωματικής αποτελεί το θεωρητικό μέρος αυτής της εργασίας. Σε αυτό περιγράφεται η έννοια της ΥΣ όπως παρουσιάζεται στην σχετική βιβλιογραφία, παρουσιάζονται οι γλώσσες οπτικού προγραμματισμού και τα εργαλεία οπτικού προγραμματισμού όπως η Blockly που χρησιμοποιούνται για την δημιουργία περιβαλλόντων ανάπτυξης ΥΣ και τέλος παρουσιάζονται εργαλεία ανάπτυξης ΥΣ που επιλέχτηκαν επειδή εφαρμόζουν κάποιο πλαίσιο ΥΣ ή γιατί έχουν συμπεριληφθεί σε έρευνες που αξιολογούν την ανάπτυξη της ΥΣ μέσω του παιδαγωγικών μεθόδων. Επιπλέον γίνεται μια σύντομη αναφορά στα παιχνίδια Blockly που αποτελούν τη βάση για την ανάπτυξη του παιχνιδιού aMazeD.

Στο τρίτο κεφάλαιο της εργασίας παρουσιάζεται η μεθοδολογία που ακολουθήθηκε για την ανάπτυξη του εκπαιδευτικού εργαλείου aMazeD και τα στάδια για την ανάπτυξη του παιχνιδιού.

Στο τέταρτο κεφάλαιο παρουσιάζεται το παιχνίδι aMazeD, τα επίπεδα του παιχνιδιού, τα εργαλεία που περιέχει και η ανάδραση/βοήθεια που προσφέρεται στις δυο εκδόσεις του παιχνιδιού.

Στο πέμπτο κεφάλαιο παρουσιάζεται η αρχιτεκτονική του παιχνιδιού aMazeD, ο τρόπος με τον οποίο συνδέονται τα αρχεία του παιχνιδιού, οι τροποποιήσεις του αρχικού κώδικα των παιχνιδιών Blockly και ο κώδικας των βασικών συναρτήσεων και λειτουργιών του παιχνιδιού που αναπτύχθηκαν για να ικανοποιηθούν οι απαιτήσεις που τέθηκαν κατά τη σχεδίαση του παιχνιδιού. Στο τέλος του 5ου κεφαλαίου συμπεριλαμβάνεται μια συνοπτική παρουσίαση της αξιολόγησης του παιχνιδιού στα πλαίσια άλλης έρευνας.

Το έβδομο κεφάλαιο περιλαμβάνει τα συμπεράσματα από την ανάπτυξη του εκπαιδευτικού εργαλείου aMazeD, αναφέρονται οι αδυναμίες και οι περιορισμοί της μελέτης και παρατίθενται κάποιες προτάσεις για την μελλοντική έρευνα και επέκταση του εργαλείου.

2. Θεωρητικό Υπόβαθρο

2.1. Εισαγωγή

Σε αυτό το κεφάλαιο αρχικά γίνεται μια εισαγωγή στην έννοια της Υπολογιστικής Σκέψης (ΥΣ) και στην επιρροή της στην εκπαίδευση. Παρουσιάζονται διάφορα πλαίσια ανάπτυξης ΥΣ και τα εργαλεία για την ανάπτυξη εφαρμογών ΥΣ. Γίνεται μια εισαγωγή στην βιβλιοθήκη της Blockly που αποτελεί το εργαλείο για την ανάπτυξη του περιβάλλοντος ΥΣ αυτής της διπλωματικής και παρουσιάζονται διάφορα περιβάλλοντα ΥΣ και τα χαρακτηριστικά τους.

2.2. Υπολογιστική σκέψη

Ο όρος υπολογιστική σκέψη χρησιμοποιήθηκε πρώτη φορά από τον Seymour Papert το 1980 στο βιβλίο του «Mindstorms: Children, computers, and powerful ideas» σε μια προσπάθεια του να θέσει τις βάσεις για ένα νέο τρόπο θεώρησης της εκπαίδευσης και αργότερα το 1996 στο άρθρο του με τίτλο "An exploration in the space of mathematics educations" στο οποίο προσδιορίζει 7 νέες διαστάσεις στην μαθηματική εκπαίδευση. Το 2006 η Jeanette Wing με το άρθρο της “Υπολογιστική Σκέψη” (Wing, 2006) έφερε στο προσκήνιο την έννοια της υπολογιστική σκέψης και δημοσιοποίησε την ιδέα της για την χρήση της ΥΣ σε όλους τους νέους φοιτητές πανεπιστημίων (Humphreys, και συν., 2015). Κατά την Wing η υπολογιστική σκέψη είναι μια βασική δεξιότητα για όλους και πρέπει να προστεθεί μαζί με την ανάγνωση, την γραφή και την αριθμητική στις αναλυτικές ικανότητες κάθε παιδιού. Επιπλέον στο άρθρο της αναφέρει ότι η υπολογιστική σκέψη περιλαμβάνει την επίλυση προβλημάτων, τον σχεδιασμό συστημάτων και την κατανόηση της ανθρώπινης συμπεριφοράς, κάνοντας χρήση των βασικών εννοιών της επιστήμης των υπολογιστών. Μερικά από τα χαρακτηριστικά της υπολογιστικής σκέψης (Wing, 2006) είναι τα εξής:

- Στηρίζει την σκέψη σε πολλά επίπεδα αφαίρεσης είναι δηλαδή εννοιολογική και όχι προγραμματιστική.
- Είναι μια βασική δεξιότητα που κάθε άνθρωπος πρέπει να γνωρίζει για να λειτουργήσει σωστά στην σύγχρονη κοινωνία.
- Είναι ένας τρόπος με τον οποίο οι άνθρωποι μπορούν να λύνουν προβλήματα. Δεν προσπαθεί να κάνει τους ανθρώπους να σκέφτονται σαν υπολογιστές.
- Συμπληρώνει και συνδυάζει την μαθηματική και την μηχανική σκέψη.
- Είναι ένα σύνολο ιδεών και υπολογιστικών εννοιών που χρησιμοποιούμε για την επίλυση προβλημάτων, την διαχείριση της καθημερινότητάς μας, την επικοινωνία

και αλληλεπίδραση με άλλους ανθρώπους. Δεν αποτελεί απλά ένα σύνολο τεχνουργημάτων που αφορούν το υλικό και το λογισμικό υπολογιστών.
- Είναι για όλους και παντού.

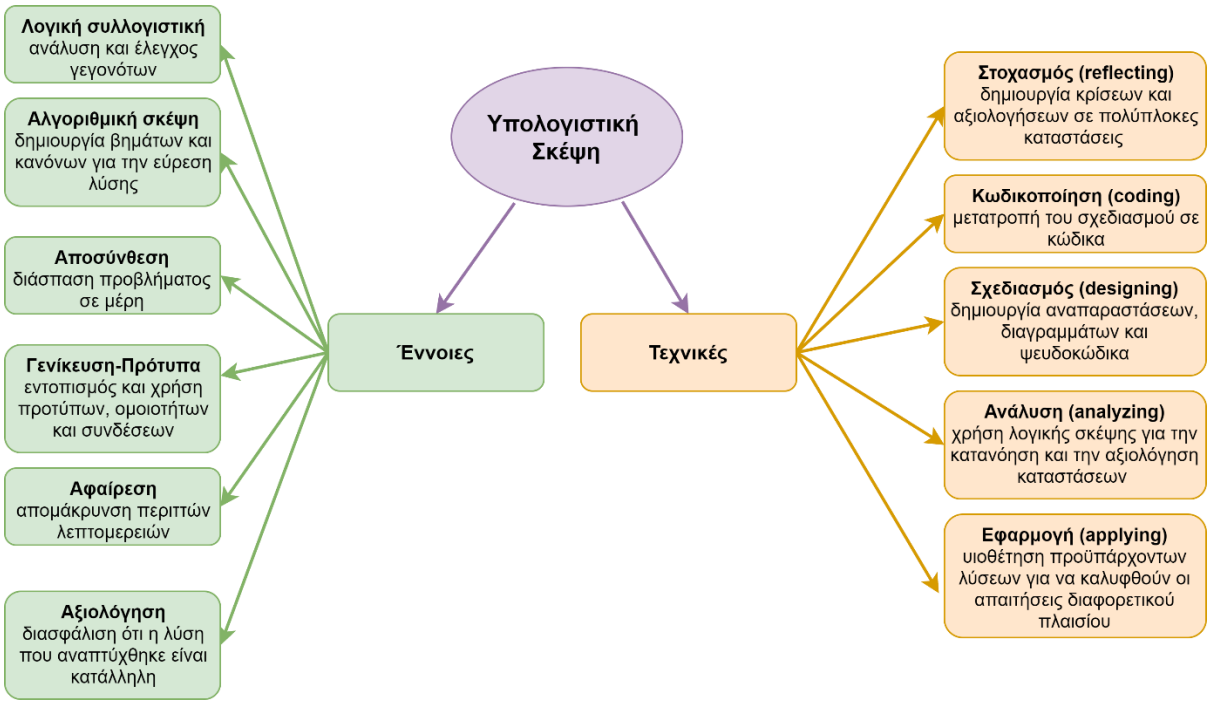
Ωστόσο υπάρχει μικρή συμφωνία σχετικά με το τι περιλαμβάνει η υπολογιστική σκέψη και το τι στρατηγικές πρέπει να ακολουθηθούν για την αξιολόγηση της ανάπτυξης της, σε νέους ανθρώπους (Brennan & Resnick, 2012). Το 2011 ο μη κερδοσκοπικός οργανισμός ISTE (International Society for Technology in Education) και η ένωση καθηγητών της Επιστήμης των Υπολογιστών CSTA (Computer Science Teachers Association) υπό την αιγίδα του NSF Αμερικής (National Science Foundation) προσπάθησαν να δημιουργήσουν ένα γενικό πλαίσιο υπολογιστικής σκέψης και να βοηθήσουν τους μαθητές να αναπτύξουν ικανότητες υπολογιστικής σκέψης (Yağcı, 2019). Σύμφωνα με τον ορισμό που δόθηκε (ISTE & CSTA, 2011) η υπολογιστική σκέψη είναι μια διαδικασία επίλυσης προβλημάτων που περιλαμβάνει:

- Την διατύπωση προβλημάτων με κατάλληλο τρόπο ώστε να είναι δυνατή η χρήση υπολογιστή και άλλων εργαλείων για την επίλυσή τους.
- Λογική οργάνωση και ανάλυση δεδομένων
- Παρουσίασή δεδομένων χρησιμοποιώντας αφαιρετικές δομές, όπως είναι τα μοντέλα και οι προσομοιώσεις.
- Αυτοματοποίηση λύσεων με την χρήση Αλγοριθμικής Σκέψης.
- Αναγνώριση, ανάλυση και υλοποίηση πιθανών λύσεων με σκοπό την επίτευξη του πιο αποτελεσματικού και εποικοδομητικού συνδυασμού βημάτων και πόρων.
- Γενίκευση και μεταφορά της συγκεκριμένης διαδικασίας επίλυσης σε μια μεγαλύτερη ποικιλία προβλημάτων.

Το 2015 ο μη κερδοσκοπικός οργανισμός CAS (Computing At School) του Ηνωμένου Βασιλείου, υπό την αιγίδα του BCS, The Chartered Institute for IT εξέδωσε έναν ηλεκτρονικό οδηγό[1] για την διδασκαλία της υπολογιστικής σκέψης στα σχολεία, στο Διάγραμμα 1, παρουσιάζουμε συνοπτικά τις έννοιες (concepts) και τις τεχνικές της υπολογιστικής σκέψης που περιγράφονται στον οδηγό αυτό.

[1] Computational Thinking-A guide for teachers (Humphreys, et al., 2015)

Διάγραμμα 1: Έννοιες και τεχνικές υπολογιστικής σκέψης[2] (Humphreys, και συν., 2015)



Σύμφωνα με τον οδηγό η υπολογιστική σκέψη είναι μια διαδικασία σκέψης που περιλαμβάνει την λογική συλλογιστική (logical reasoning), με την οποία προβλήματα μπορούν να λυθούν και τεχνουργήματα, διαδικασίες και συστήματα μπορούν να κατανοηθούν σε μεγαλύτερο βαθμό. Περιλαμβάνει τις ικανότητες του να σκέφτεσαι

- αλγοριθμικά
- ως προς την αποσύνθεση ενός προβλήματος
- ως προς την γενίκευση και την αναγνώριση και χρήση προτύπων
- αφαιρετικά, διαλέγοντας κατάλληλες αναπαραστάσεις και
- αξιολογικά

2.3. Εργαλεία Ανάπτυξης Περιβάλλοντος Υπολογιστικής Σκέψης

Με τον όρο εργαλεία ανάπτυξης υπολογιστικής σκέψης αναφερόμαστε σε γλώσσες προγραμματισμού, σε γλώσσες και περιβάλλοντα οπτικού προγραμματισμού και στις πλατφόρμες που διατίθενται για την ανάπτυξη ψηφιακών παιχνιδιών από μαθητές.

[2] Το διάγραμμα σχεδιάστηκε με την χρήση του app.diagrams.net

2.3.1. Γλώσσες και Περιβάλλοντα Οπτικού Προγραμματισμού

Μια γλώσσα προγραμματισμού χρησιμοποιείται για να γραφτεί ένα πρόγραμμα ή ένα σύνολο οδηγιών. Ουσιαστικά μια γλώσσα προγραμματισμού είναι ένα σύνολο από εντολές με την μορφή δεσμευμένων λέξεων (π.χ. do, if, while) που παράγουν αποτελέσματα σε γλώσσα μηχανής, τη γλώσσα δηλαδή που καταλαβαίνει ο υπολογιστής. Ως γλώσσα οπτικού προγραμματισμού ορίζουμε οποιαδήποτε γλώσσα προγραμματισμού επιτρέπει την δημιουργία προγραμμάτων χρησιμοποιώντας γραφικό χειρισμό των εντολών της (visual based), αντί για εντολές γραμμένες σε μορφή κειμένου (text based). Οι γλώσσες οπτικού προγραμματισμού προσφέρουν ένα περιβάλλον ανάπτυξης προγραμμάτων (IDE) ή ανάπτυξη των προγραμμάτων εξολοκλήρου στον ιστό. Και στις δυο περιπτώσεις συνθήκες, εντολές και δομές ελέγχου παρουσιάζονται με την μορφή μπλοκ και εικονιδίων, τα οποία μπορούν να συνδυαστούν για την δημιουργία προγραμμάτων. Οι γλώσσες οπτικού προγραμματισμού έχουν ένα μεγάλο φάσμα εφαρμογών και χρησιμοποιούνται σε ποικίλους τομείς όπως η εκπαίδευση, τα πολυμέσα, η ανάπτυξη παιχνιδιών, ο αυτοματισμός, η διαχείριση δεδομένων και η προσομοίωση συστημάτων.

Στον παρακάτω πίνακα παρουσιάζουμε τις πιο δημοφιλείς γλώσσες που χρησιμοποιούνται στον τομέα της εκπαίδευσης και δίνουμε μια σύντομη περιγραφή τους.

Πίνακας 1: Γλώσσες Οπτικού Προγραμματισμού στην Εκπαίδευση

Γλώσσα	Περιγραφή	
Alice	**Δημιουργός**	University of Virginia, Carnegie Mellon University
	Ιστοσελίδα	http://www.alice.org/
	Χαρακτηριστικά	Γλώσσα οπτικού προγραμματισμού με ολοκληρωμένο περιβάλλον ανάπτυξης (IDE). Χρησιμοποιείται για την ανάπτυξη μικρών παιχνιδιών και δια δραστικών κινούμενων ιστοριών σε 3D περιβάλλον. Συνδυάζει την ανάπτυξη αντικειμενοστραφούς κώδικα με την χρήση εννοιών όπως είναι οι μέθοδοι και οι μέθοδοι ανίχνευσης γεγονότος (event listeners), κλάσεις με την μορφή σκηνών και αντικείμενα με την μορφή χαρακτήρων.

	Στοιχεία την που ξεχωρίζουν	• Διδασκαλία του προγραμματισμού και της θεωρίας του χωρίς σημασιολογία. • Με τη χρήση του IDE δεν χρειάζεται η απομνημόνευση συντακτικών κανόνων. • Απευθύνεται μαθητές και ενθαρρύνει την αφήγηση ιστοριών. • Σε συνδυασμό με το λογισμικό NetBeans μπορεί να μετατραπούν τα αρχεία Alice σε κώδικα Java.
Kodu Game Lab (KGL)	**Δημιουργός**	Microsoft's FUSE Labs
	Ιστοσελίδα	https://www.kodugamelab.com/
	Χαρακτηριστικά	Η KGL με το IDE της Kodu, αποτελεί ένα ολοκληρωμένο περιβάλλον οπτικού προγραμματισμού για την ανάπτυξη τρισδιάστατων παιχνιδιών σε Windows και Xbox. Ο χρήστης μπορεί να τροποποιήσει τον Kodu-κόσμο με ένα οπτικό σύστημα επεξεργασίας (το οποίο είναι προσβάσιμο και με χειριστήριο) αποτελούμενο από έδαφος και 3D αντικείμενα και να δημιουργήσει χαρακτήρες οι οποίοι θα εκτελούν τον κώδικα του χρήστη. Η KGL είναι εστιασμένη στα γεγονότα (event-driven), όταν συμβαίνει ένα γεγονός απαιτείται μια δράση.
	Στοιχεία που την ξεχωρίζουν	• Διδασκαλία κώδικα με την χρήση μπλοκ και εικόνων. • Δημιουργικότητα, επίλυση προβλημάτων, αφήγηση ιστοριών • Ανάπτυξη τρισδιάστατων κόσμων μέσω ενός εύχρηστου περιβάλλοντος.
Scratch	**Δημιουργός**	MIT Media Lab
	Ιστοσελίδα	https://scratch.mit.edu/
	Χαρακτηριστικά	Γλώσσα οπτικού προγραμματισμού υψηλού επιπέδου, χρησιμοποιεί μπλοκ για την δημιουργία προγραμμάτων. Η δημιουργία, επεξεργασία και αποθήκευση έργων γίνεται

		εξολοκλήρου διαδικτυακά στον ιστό και επιπλέον υπάρχει μεγάλη κοινότητα (Scratchers) με την οποία ο χρήστης μπορεί να μοιραστεί έργα ή να βασιστεί σε έργα άλλων, να τα επεξεργαστεί και να δημιουργήσει πιο σύνθετα προγράμματα. Επειδή ήταν από τα πρώτα εργαλεία που αναπτύχθηκαν έχει επηρεάσει σημαντικά όλες τις άλλες γλώσσες.
	Στοιχεία που την ξεχωρίζουν	• Δημιουργία μιας πιο εύχρηστης γλώσσας για μαθητές. • Δημιουργία έργων με νόημα για τον δημιουργό, μέσω της διεπαφής της γλώσσας. • Χρήση μπλοκ που αλληλοσυνδέονται κατάλληλα για την δημιουργία σύνθετων προγραμμάτων. • Κοινωνικότητα των μαθητών μέσω της αλληλεπίδρασης και της συνεργασίας που προσφέρει η διαδικτυακή κοινότητα του Scratch.
Snap!	**Δημιουργός**	Brian Harvey, Jens Mönig UC Berkeley
	Ιστοσελίδα	https://snap.berkeley.edu/
	Χαρακτηριστικά	Γλώσσα οπτικού προγραμματισμού εξολοκλήρου εκτελέσιμη στον ιστό. Είναι επηρεασμένη από την γλώσσα Scratch. Προσφέρει τις εξής βελτιώσεις: η δομή της στηρίζεται σε HTML και JavaScript επομένως δες εξαρτάται από τον Flash και δεν υπάρχει περιορισμός στις πλατφόρμες που χρησιμοποιείται. Επιπλέον σου επιτρέπει να δημιουργείς δικά σου μπλοκ, χρησιμοποιεί τύπους δεδομένων πρώτης τάξης, όπως η τιμή που επιστρέφεται από μια διαδικασία, και σου παρέχει την δυνατότητα να διαχειρίζεσαι συνεδρίες με πολλαπλούς χρήστες σε πραγματικό χρόνο.

	Στοιχεία που την ξεχωρίζουν	• Διδασκαλία του προγραμματισμού σε μαθητές με χρήση μπλοκ (παρέχονται 8 διαφορετικές κατηγορίες μπλοκ). • Χρήση των μπλοκ όχι μόνο ως στοιχεία ελέγχου προγραμμάτων αλλά και ως δεδομένα. • Εξαγωγή κώδικα σε Python, JavaScript, C .

2.3.2. Η βιβλιοθήκη της Blockly

Για την ανάπτυξη του περιβάλλοντος υπολογιστικής σκέψης που παρουσιάζεται σε αυτή την διπλωματική εργασία, χρησιμοποιήθηκε η βιβλιοθήκη Blockly. Με τον όρο βιβλιοθήκη αναφερόμαστε σε μια συλλογή πόρων που χρησιμοποιούνται για την ανάπτυξη λογισμικού. Η συλλογή αυτή μπορεί να περιέχει δεδομένα διαμόρφωσης, τεκμηρίωση, δεδομένα βοήθειας, έτοιμα πρότυπα όπως πρότυπα εμφάνισης μηνυμάτων, έτοιμο κώδικα και υπορουτίνες, κλάσεις , τιμές και προσδιορισμένους τύπους μεταβλητών.
Η Blockly δεν αποτελεί γλώσσα οπτικού προγραμματισμού. Είναι μια βιβλιοθήκη της JavaScript που χρησιμοποιείται για να την δημιουργία συντακτών οπτικού προγραμματισμού αλλά και την ανάπτυξη γλωσσών οπτικού προγραμματισμού. Η ανάπτυξη της Blockly ξεκίνησε το 2012 από την Google και το MIT, με υπεύθυνο ανάπτυξης τον Neil Fraser και συνεισφερόμενους προγραμματιστές τους Quynh Neutron, Ellen Spertus και Mark Friedman.

Εικόνα 1: Λογότυπο Blockly

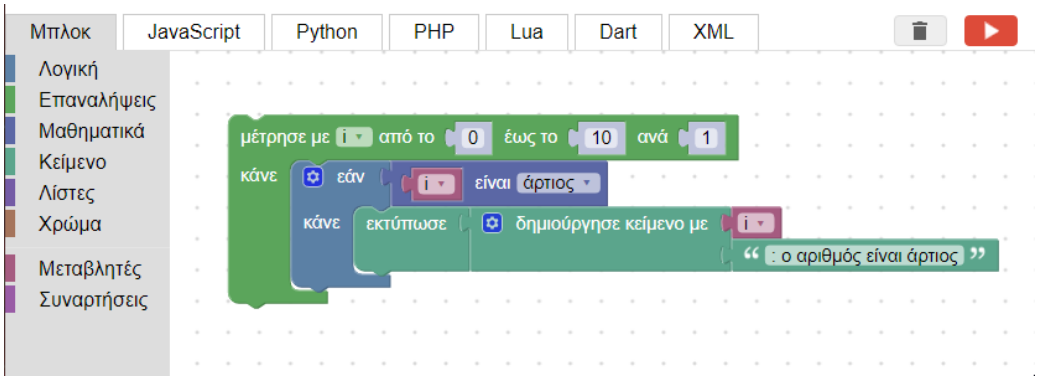
Χαρακτηριστικά της Blockly και πλεονεκτήματα

Από την οπτική του χρήστη η Blockly είναι ένα περιβάλλον οπτικού προγραμματισμού, ένας εύκολος τρόπος για την δημιουργία κώδικά χρησιμοποιώντας μπλοκ (blocks). Στην

ουσία ο χρήστης δεν κάνει χρήση της βιβλιοθήκης αλλά των εφαρμογών που έχουν αναπτυχθεί με βάση την Blockly και περιέχουν τον συντάκτη της Blockly.

Ο συντάκτης της Blockly χρησιμοποιεί αλληλοσυνδεόμενα μπλοκ (block-based) για την δημιουργία κώδικα, με αυτόν τον τρόπο γίνεται πιο ελκυστική η εκμάθηση προγραμματισμού σε μικρές ηλικίες και αρχάριους καθώς απαλείφονται τα συντακτικά λάθη και η μονοτονία της συγγραφής κώδικα μέσω κειμένου (text-based).

Εικόνα 2: Συντάκτης της Blockly (Blockly editor)



Η διεπαφή χρήστη του συντάκτη της Blockly αποτελείται από την εργαλειοθήκη (toolbox), τον χώρο εργασίας (workspace) στον οποίο ο χρήστης μπορεί να τοποθετεί τα μπλοκ που επιλέγει από την εργαλειοθήκη και ένα κουμπί για την εκτέλεση του κώδικα ( ).

Ο κώδικας της εικόνας 2 εκτυπώνει ως ειδοποίηση (alert window) μόνο τους άρτιους αριθμούς από το 0 ως το 10.

Εικόνα 3: Εκτέλεση του κώδικα εικόνας 2

Ένα από τα πιο σημαντικά χαρακτηριστικά της Blockly είναι η δυνατότητα της να εξάγει τον κώδικα που δημιούργησε ο χρήστης με την βοήθεια των μπλοκ σε συντακτικά σωστό κώδικα των υποστηριζόμενων γλωσσών προγραμματισμού. Οι γλώσσες που υποστηρίζει η Blockly είναι οι JavaScript, Python , PHP, Lua και Dart.

Εικόνα 4: Κώδικας για τα μπλοκ της εικόνας 2 σε γλώσσα JavaScript

```
var i;


for (i = 0; i <= 10; i++) {
  if (i % 2 == 0) {
    window.alert(String(i) + ': ο αριθμός είναι άρτιος');
  }
}
```

Οι δημιουργοί της Blockly έδωσαν μεγάλη έμφαση στον γραφικό σχεδιασμό, σύμφωνα με τον Neil Fraser (2012) για τα περιβάλλοντα οπτικού προγραμματισμού «Οι χρήστες βυθίζονται σε ένα άγνωστο περιβάλλον και η εμφάνιση και η αίσθηση μπορούν να τους κάνουν να αποφασίσουν ότι "δεν τους αρέσει". Η Blockly έχει ένα εξαιρετικά προσεγμένο περιβάλλον χρήστη: θαμπάδες, στρογγυλεμένες γωνίες και ανατροφοδότηση μέσω ήχων.»[3] Επιπλέον η Blockly είναι μεταφρασμένη σε περισσότερες από 40 γλώσσες βοηθώντας στην άμεση εξοικείωση του χρήστη με το περιβάλλον ασχέτως από το αν γνωρίζει αγγλικά ή όχι. Από την οπτική του προγραμματιστή/ σχεδιαστή (developer) μιας εφαρμογής η Blockly είναι μια έτοιμη διεπαφή χρήστη (UI) που εξάγει συντακτικά σωστό κώδικα από τον κώδικα που δημιουργεί ο χρήστης με την βοήθεια των μπλοκ. Είναι εξολοκλήρου εκτελέσιμη στην πλευρά του πελάτη (client-side) και δεν υπάρχουν εξαρτήσεις από εξυπηρετητή, επιπλέον είναι συμβατή με τους πιο δημοφιλείς φυλλομετρητές ιστού όπως Chrome , Firefox, Safari, Opera και IE. Τέλος υποστηρίζει την ανάπτυξη εφαρμογών σε περιβάλλοντα Android, iOS. Η Blockly είναι ένα από τα πολλά περιβάλλοντα οπτικού προγραμματισμού, σίγουρα δεν μπορεί να καλύψει τις ανάγκες όλων των εφαρμογών και πρέπει να γίνει εκτενής έρευνα για να επιλεχθεί το κατάλληλο περιβάλλον που θα χρησιμοποιηθεί σε μια εφαρμογή. Παραθέτουμε τα βασικότερα πλεονεκτήματα της Blockly σύμφωνα με την επίσημη ιστοσελίδα της βιβλιοθήκης για προγραμματιστές[4]:

- **Δυνατότητα εξαγωγής του κώδικα :** Οι χρήστες μπορούν να εξάγουν τον δομημένο με μπλοκ κώδικά τους σε κώδικα κειμένου μιας εκ των υποστηριζόμενων γλωσσών προγραμματισμού της Blockly. Με αυτή την δυνατότητα μπορούν ομαλά να μεταβούν στην συγγραφή κώδικα με κειμενογράφο.

[3] https://neil.fraser.name/news/2012/06/19/

[4] https://developers.google.com/blockly/guides/overview : Blockly’s strengths and other options

- **Βιβλιοθήκη ανοιχτού Κώδικα:** Οτιδήποτε σχετίζεται με την Blockly είναι ανοιχτού κώδικα και μπορεί να αντιγραφεί , να τροποποιηθεί και να εισαχθεί σε προσωπικές εφαρμογές και ιστοσελίδες.
- **Επεκτάσιμη Βιβλιοθήκη:** Μπορεί να τροποποιηθεί και να αναπρογραμματιστεί ανάλογα με τις ανάγκες του κάθε σχεδιαστή. Ο σχεδιαστής/προγραμματιστής μπορεί να προσθέσει προσαρμοσμένα μπλοκ στην διεπαφή της εφαρμογής του (API) ή να αφαιρέσει μπλοκ που δεν έχουν χρηστικότητα και δεν παρέχουν κάποια λειτουργικότητα στην εφαρμογή του.
- **Υψηλές Ικανότητες:** Η Blockly δεν είναι παιχνίδι. Ο σχεδιαστής / προγραμματιστής μπορεί να εκτελέσει πολύπλοκες υπολογιστικές εργασίες όπως ο υπολογισμός τυπικής απόκλισης με ένα και μόνο μπλοκ.
- **Διεθνής:** Η Blockly υποστηρίζει την μετάφραση των μπλοκ, των μηνυμάτων βοήθειας και γενικότερα όλων των κειμένων μιας διεπαφής σε περισσότερες από 40 γλώσσες.

Η Blockly έχει επηρεάσει στο έπακρο τις γλώσσες και τα περιβάλλοντα οπτικού προγραμματισμού, πολλά δημοφιλή έργα βασίζονται στην Blockly και έχουν δημιουργηθεί με βάση την βιβλιοθήκη.

Εικόνα 5: Λογότυπα έργων που αναπτύχθηκαν με την Blockly

Τεκμηρίωση

Η επίσημη ιστοσελίδα της Blockly περιέχει εκτενείς οδηγούς για την εκμάθηση της βιβλιοθήκης και την χρήση των στοιχείων της. Οι πληροφορίες αφορούν κυρίως τις παρακάτω κατηγορίες:

- Τρόποι εισαγωγής της βιβλιοθήκης (Injection)
- Χώρος εργασίας (Workspace)

- Εργαλειοθήκη (Toolbox)
- Γεννήτρια Κώδικα (Code Generator)
- Συμβάντα (Events)

Στο Κεφάλαιο 5 γίνεται η χρήση των παραπάνω οδηγών για την υλοποίηση του εκπαιδευτικού παιχνιδιού aMazeD.

2.4. Περιβάλλοντα Ανάπτυξης Υπολογιστικής Σκέψης και η χρήση τους σε διδακτικές μεθόδους

Με τον όρο περιβάλλοντα ανάπτυξης υπολογιστικής σκέψης αναφερόμαστε σε παιχνίδια και εφαρμογές εκπαιδευτικού σκοπού που χρησιμοποιούν τα εργαλεία οπτικού προγραμματισμού ή και κλασικές γλώσσες προγραμματισμού για την ανάπτυξη της υπολογιστικής σκέψης στους χρήστες τους.

Με την εξέλιξη της τεχνολογίας και την εισαγωγή νέων εκπαιδευτικών μεθόδων στα σχολεία η χρήση εργαλείων οπτικού προγραμματισμού και η δημιουργία απλών ψηφιακών παιχνιδιών προτάθηκαν ως πλαίσια εκμάθησης υπολογιστικής σκέψης (Kazimoglu, Kiernan, Bacon, & MacKinnon, 2012). Αξιοσημείωτη είναι η συνεισφορά του κινήματος Hour of Code[5] (hourofcode.com) που ξεκίνησε από τους ιδρυτές του μη κερδοσκοπικού οργανισμού Code.org με στόχο την ανάπτυξη της υπολογιστικής σκέψης σε νεαρές ηλικίες. Το κίνημα αυτό προτείνει απλές ψηφιακές δραστηριότητες με την μορφή προγραμματισμού που μπορούν να ολοκληρωθούν σε μια ώρα, βασιζόμενες κυρίως σε εκπαιδευτικά παιχνίδια που χρησιμοποιούν την βιβλιοθήκη της Blockly. Με την συμμετοχή στο κίνημα πολλών γνωστών εταιρειών όπως Microsoft, Apple, Amazon και προσωπικοτήτων όπως ο Mark Zuckerberg και ο Bill Gates έγινε ένα παγκόσμιο κίνημα που προσφέρει δραστηριότητες σε περισσότερες από 180 χώρες, μεταφρασμένες σε περισσότερες από 45 γλώσσες.

Για την ανάπτυξη του εργαλείου που παρουσιάζεται σε αυτή την εργασία έγινε έρευνα σε πολλά διαθέσιμα εκπαιδευτικά παιχνίδια και στον τρόπο με τον οποίο συνδέονται με πλαίσια ανάπτυξης Υπολογιστικής σκέψης.

[5] Στα ελληνικά είναι γνωστό ως Η Ώρα του Κώδικα

2.4.1. Εφαρμογή πλαισίου Υπολογιστικής Σκέψης - Το παράδειγμα του Program Your Robot

Οι Kazimoglu, Kiernan, Bacon, & MacKinnon, το 2012 παρουσίασαν ένα πλαίσιο δεξιοτήτων υπολογιστικής σκέψης και την εφαρμογή του στο εκπαιδευτικό εργαλείο Program Your Robot (https://cainkazimoglu.com/program-your-robot), που σχεδιάστηκε και αναπτύχθηκε από τους ίδιους. Παρόλο που το παιχνίδι δεν είναι πλέον διαθέσιμο για χρήστες η μελέτη των Kazimoglu, Kiernan, Bacon, & MacKinnon είναι ουσιαστική για την δημιουργία ενός εργαλείου υπολογιστικής σκέψης και την εφαρμογή πλαισίου σε αυτό.

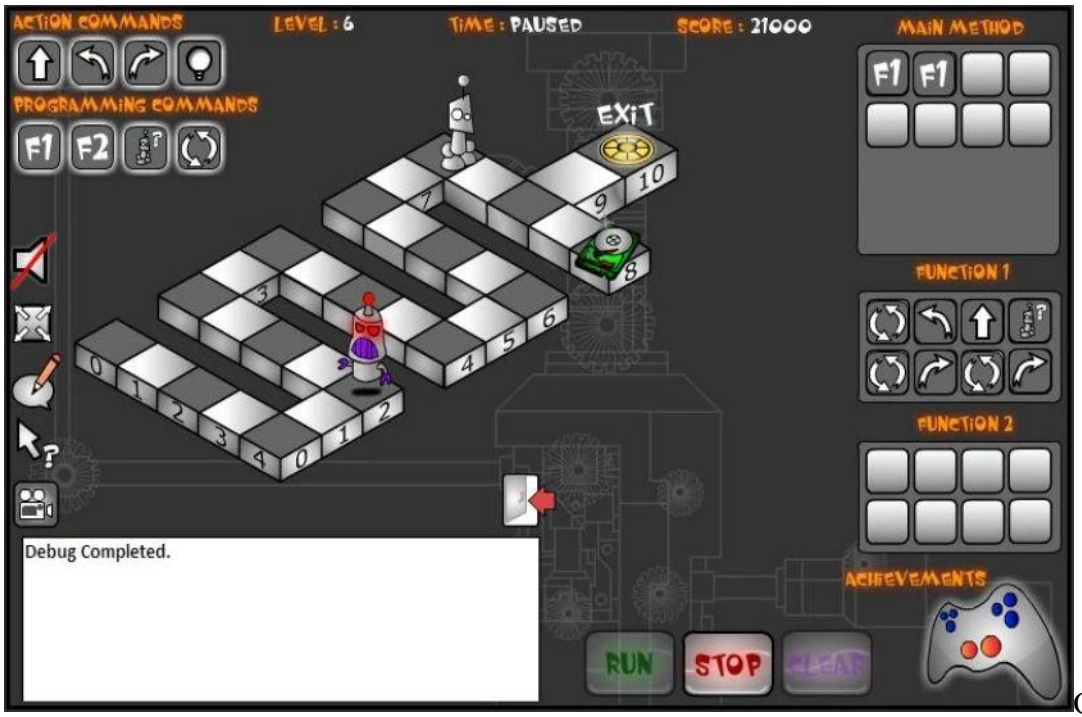
Το Program Your Robot ήταν ένα εκπαιδευτικό παιχνίδι στο οποίο ο χρήστης μπορούσε να ελέγξει τον χαρακτήρα του (ένα ρομπότ) δίνοντας του εντολές με την χρήση εικονιδίων που προσομοιώνουν κίνηση. Στόχος του χρήστη ήταν να οδηγήσει το ρομποτ μέσα από το διαθέσιμο μονοπάτι στον τηλεμεταφορέα (τέλος μονοπατιού) και να ανάψει τα φώτα του. Στην πορεία του μονοπατιού έπρεπε να αποφύγει εχθρικά ρομπότ και να συλλέξει τα διαθέσιμα ανά επίπεδο αντικείμενα, ενώ έπρεπε να ολοκληρώσει το πρόβλημα του κάθε επιπέδου στον διαθέσιμο χρόνο. Το παιχνίδι επιβράβευε τον χρήστη με την μορφή σκορ για την επιτυχή ολοκλήρωση του επιπέδου, την αποφυγή εχθρικών ρομποτ και την συλλογή αντικειμένων.

Ο χρήστης είχε στην διάθεσή του δυο κατηγορίες εντολών (εντολές δράσης και εντολές προγραμματισμού) και μπορούσε να κατασκευάσει συναρτήσεις (function 1 & 2) με αλληλουχίες βημάτων και να τις χρησιμοποιήσει στο κυρίως πρόγραμμά του (main method). Μπορούσε να κάνει χρήση των εντολών αυτών σέρνοντας τα διαθέσιμα εικονίδια (drag and drop) στις επιτρεπόμενες θέσεις (slots). Επιπλέον είχε την δυνάτοτητα να τρέξει τον κώδικα του και να εντοπίσει πιθανά λάθη μέσω των πλήκτρων Run, Stop, Clear. Για την επιτυχή ολοκλήρωση των επιπέδων ο χρήστης έκανε χρήση ακολουθιών, συνθηκών, βρόχων επανάληψης και μεθόδων.

Υπήρχαν δυο βασικοί τύποι κανόνων στο παιχνίδι:

- Λειτουργικοί κανόνες: παρουσιάζονται με την μορφή οδηγιών σε πλαίσια διαλόγου στην αρχή κάθε επιπέδου που ενημερώνουν τον χρήστη για τα χαρακτηριστικά του παιχνιδιού και πως λειτουργούν οι εντολές.
- Ακολουθιακοί (Consecutive) κανόνες: άγραφοι κανόνες που υπονοούνται στο παιχνίδι. Αποτελούν διαδικασίες ανάπτυξης αποτελεσματικών στρατηγικών για την επιτυχή ολοκλήρωση του παιχνιδιού.

Εικόνα 6: Program Your Robot



Οι ακολουθιακοί κανόνες παρουσιάζονται μέσα από το βαθμολογικό σύστημα του παιχνιδιού, το οποίο συνδέεται άμεσα με την ικανότητα του παίκτη να κατανοεί τους κανόνες του παιχνιδιού, και να αναπτύσσει κατάλληλες στρατηγικές προσαρμόζοντας την συμπεριφορά του ρομποτ στους κανόνες αυτούς. Για να επιτευχθεί αυτό η βαθμολογία του παίκτη εξαρτάται από το πλήθος των αντικειμένων που συλλέγει ο παίκτης, τις εντολές προγραμματισμού που χρησιμοποιεί και το πλήθος των slots που χρησιμοποιεί. Το βαθμολογικό σύστημα όπως δομήθηκε στο παιχνίδι αποτελεί ένα κίνητρο για να κατασκευάσουν οι χρήστες στρατηγικές που θα τους οδηγήσουν στην νίκη, ενθαρύννοντας την αξιολόγηση των στρατηγικών αυτών ως προς την αποδοτικότητα τους.

Στην έρευνα τους (Kazimoglu, Kiernan, Bacon, & MacKinnon, 2012) επιχειρηματολογούν ότι υπάρχουν 5 βασικές δεξιότητες που χαρακτηρίζουν την υπολογιστική σκέψη

- **Επίλυση Προβλημάτων** (Problem Solving)
- **Δημιουργία Αλγορίθμων** (Building Algorithms)
- **Εντοπισμός Σφαλμάτων** (Debugging)
- **Προσομοίωση** (Simulation)
- **Κοινωνικοποίηση** (Socializing)

Η εφαρμογή του παραπάνω πλαισίου στο παιχνίδι φαίνεται στον Πίνακας 2. Το παιχνίδι αξιολογήθηκε από μαθητές Λυκείου και πρωτοετείς φοιτητές Πανεπιστημίου και η πλειοψηφία των ερωτηθέντων βρήκε το παιχνίδι κατάλληλο να βοηθήσει τους χρήστες του

στην εισαγωγή προγραμματιστικών κατασκευών και στην απόκτηση ικανοτήτων υπολογιστικής σκέψης σύμφωνα με το προτεινόμενο πλαίσιο.

Πίνακας 2: Παραδείγματα δραστηριοτήτων του παιχνιδιού Program Your Robot και η συσχέτιση τους με διάφορα χαρακτηριστικά της Υπολογιστικής Σκέψης, πηγή: Kazimoglu, Kiernan, Bacon, & MacKinnon, 2012

Εργασία	Συσχετιζόμενες ικανότητες ΥΣ	Δραστηριότητα του παιχνιδιού
Αναγνώριση και αποσύνθεση προβλήματος	Επίλυση Προβλημάτων	Βοήθησε το ρομπότ να φτάσει στον τηλεμεταφορέα. Ενεργοποίησε τα φώτα του ρομπότ όταν βρίσκεται πάνω στον τηλεμεταφορέα.
Δημιουργία αποδοτικών και επαναλαμβανόμενων προτύπων	Δημιουργία Αλγορίθμων	Δημιούργησε έναν αλγόριθμο που θα επιλύει το κάθε επίπεδο με την χρήση όσων των δυνατόν λιγότερων slots. Κάνε χρήση συναρτήσεων για επαναλαμβανόμενα πρότυπα.
Εξάσκηση στον εντοπισμό σφαλμάτων (debug-mode)	Εντοπισμός Σφαλμάτων	Πάτησε το πλήκτρο RUN για να παρακολουθήσεις τον αλγόριθμο της λύσης σου και να ανακαλύψεις πιθανά λάθη στην λογική της.
Εξάσκηση στην εκτέλεση του κώδικα (run-time mode)	Προσομοίωση	Παρατήρησε τις κινήσεις του ρομπότ κατά την διάρκεια που εκτελείται το πρόγραμμά σου. Μπορείς να ακολουθήσεις τον αλγόριθμο της λύσης σου; Παρατηρείς τις αναμενόμενες συμπεριφορές;
Ανταλλαγή Ιδεών (brainstorming)	Κοινωνικοποίηση	Εξέτασε της νικητήριες στρατηγικές άλλων παικτών. Σύγκρινε τις λύσεις τους με τις δικές σου. Τι συμβουλή θα έδινες στον εαυτό σου και σε αυτούς για την επίτευξη καλύτερης βαθμολογίας; Συζήτησε.

2.4.2. Αξιολόγηση εκπαιδευτικών εφαρμογών που αναπτύχθηκαν χρησιμοποιώντας το περιβάλλον Scratch – Πλαίσιο ΥΣ Brennan & Resnick

Έχουμε ήδη αναφερθεί στην γλώσσα προγραμματισμού Scratch και στο περιβάλλον που παρέχει για την ανάπτυξη εφαρμογών και πολυμέσων από μαθητές. Σε αυτή την παράγραφο δεν παρουσιάζουμε κάποιο εκπαιδευτικό παιχνίδι αλλά αναφέρουμε το πλαίσιο υπολογιστικής σκέψης που εισήγαγαν οι Brennan και Resnick το 2012 για την αξιολόγηση της ανάπτυξης της υπολογιστικής σκέψης σε νέους που χρησιμοποιούν την πλατφόρμα του Scratch για την σχεδίαση εφαρμογών. Σύμφωνα με αυτό το πλαίσιο υπάρχουν τρεις διαστάσεις της υπολογιστικής σκέψης:

1. **Υπολογιστικές Έννοιες**, οι έννοιες που οι μαθητές χρησιμοποιούν καθώς προγραμματίζουν.
2. **Υπολογιστικές Πρακτικές**, οι πρακτικές που οι μαθητές αναπτύσσουν καθώς προγραμματίζουν.
3. **Υπολογιστικές Αντιλήψεις**, οι αντιλήψεις που σχηματίζουν οι μαθητές για τον κόσμο γύρω τους και για τους εαυτούς τους.

Πίνακας 3: Παρουσίαση πλαισίου Brennan-Resnick (Brennan & Resnick, 2012)

Διάσταση	Έννοιες/Πρακτικές/Αντιλήψεις	Περιγραφή
Υπολογιστικές Έννοιες	Ακολουθίες (sequences)	Δραστηριότητες που εκφράζονται ως μια σειρά ανεξάρτητων βημάτων ή οδηγιών.
	Βρόγχοι Επανάληψης (loops)	Ο μηχανισμός με τον οποίο μια ακολουθία μπορεί να εκτελεστεί περισσότερες από μια φορές.
	Παραλληλισμός (parallelism)	Η ταυτόχρονη εκτέλεση ανεξάρτητων ακολουθιών.
	Γεγονότα (events)	Μια ενέργεια που έχει ως αποτέλεσμα την εκτέλεση μιας άλλης ενέργειας , για παράδειγμα την εκτέλεση μιας ακολουθίας.
	Συνθήκες (conditionals)	Η ικανότητα του να λαμβάνεις αποφάσεις βασισμένος σε συγκεκριμένες συνθήκες.
	Τελεστές (operators)	Αριθμητικοί (π.χ. +,-, mod), Λογικοί (π.χ. and, or, not) και Αλφαριθμητικοί (π.χ. join, length of) τελεστές για τον χειρισμό μαθηματικών, λογικών

		και αλφαριθμητικών εκφράσεων.
	Δεδομένα (data)	Τα δεδομένα περιλαμβάνουν την αποθήκευση, ανάκτηση και τροποποίηση τιμών. Για παράδειγμα η χρήση μεταβλητών και λιστών.
Υπολογιστικές Πρακτικές	Σταδιακή εξέλιξη και επανάληψη βημάτων (being incremental and iterative)	Ο σχεδιασμός και η υλοποίηση ενός έργου είναι μια εξελικτική διαδικασία. Αποτελείται από επαναληπτικούς κύκλους σχεδιασμού, ανάπτυξης και ελέγχου του προγράμματος και περεταίρω ανάπτυξης του, με βάση των εμπειριών που αποκτήθηκαν και νέων ιδεών.
	Έλεγχος και εντοπισμός σφαλμάτων (testing and debugging)	Ανάπτυξη στρατηγικών για την αντιμετώπιση και την πρόβλεψη προβλημάτων στην ανάπτυξη ενός έργου.
	Επαναχρησιμοποίηση (reusing and remixing)	Επαναχρησιμοποίηση, τροποποίηση και ανάμειξη έργων που δημιουργήθηκαν από άλλους με στόχο τη δημιουργία ενός πιο πολύπλοκου έργου που δεν θα ήταν δυνατό διαφορετικά.
	Αφαίρεση και σπονδυλωτή διαμόρφωση (abstraction and modularizing)	Η δημιουργία ενός μεγάλου έργου με τον συνδυασμό συλλογών από μικρότερα τμήματα. Για παράδειγμα η τμηματοποιήση κώδικα ανάλογα με την λειτουργικότητα των εντολών.
Υπολογιστικές Αντιλήψεις	Έκφραση (expressing)	Η χρήση της τεχνολογίας όχι μόνο ως καταναλωτής αλλά ως μέσο σχεδιασμού και έκφρασης.
	Σύνδεση (connecting)	Αλληλεπίδραση με άλλα άτομα.
	Προβληματισμοί/Ανησυχίες (questioning)	Η αναζήτηση δεικτών ότι οι νέοι άνθρωποι μπορούν και ενθαρρύνονται να κάνουν ερωτήσεις και να αμφισβητούν τα αυτονόητα, σε ορισμένες περιπτώσεις, απαντώντας στις ερωτήσεις αυτές με δικές τους προτάσεις και σχέδια.

Για την αξιολόγηση της ανάπτυξης της ΥΣ με βάση του πλαισίου τους, οι Brennan και Resnick ακολούθησαν τρεις προσεγγίσεις:

1) Ανάλυση του προφίλ των μαθητών: με την χρήση του εργαλείου ανάλυσης Scrape, αναλύθηκε η δομή των μπλοκ που περιέχονται στα έργα διαφόρων μαθητών. Η προσέγγιση αυτή περιορίζεται μόνο στα έργα που έχουν ανεβάσει οι μαθητές στο διαδικτυακό προφίλ τους, στην κοινότητα του Scratch.
2) Συνεντεύξεις: διεξαγωγή συνεντεύξεων σε 31 μαθητές, ηλικίας 8-17, από διαφορετικές γεωγραφικές περιοχές, διαφορετικού επιπέδου στον προγραμματισμό με Scratch (αρχάριοι ως ειδικοί) από τους οποίους το 40% ήταν γυναίκες. Οι συνεντεύξεις είχαν διάρκεια 60-120 λεπτών και περιείχαν ερωτήσεις για τις πρακτικές που χρησιμοποιεί ο μαθητής, τα έργα που αναπτύσσει, τις δραστηριότητές του στην διαδικτυακή κοινότητα του Scratch και τις προσδοκίες που έχει ενώ είχε γίνει η ανάλυση του προφίλ τους.
3) Σενάρια σχεδιασμού: σε συνεργασία με το NSF και το EDC (Education Development Center) αναπτύχθηκαν 3 συλλογές έργων Scratch με αυξανόμενη πολυπλοκότητα. Κάθε συλλογή είχε δυο έργα που αναφερόταν στις ίδιες έννοιες και πρακτικές αλλά διέφεραν αισθητικά για να καλύψουν διαφορετικά ενδιαφέροντα. Οι συλλογές παρουσιάστηκαν ξεχωριστά σε μαθητές στα πλαίσια 3 συνεντεύξεων και σε περιβάλλον τάξης, με την πληροφορία ότι δημιουργήθηκαν από έναν νεαρό μαθητή στο περιβάλλον Scratch. Από τους μαθητές ζητήθηκε να εξηγήσουν την λειτουργικότητα του επιλεγμένου έργου, να περιγράψουν πως θα μπορούσε να επεκταθεί, να διορθώσουν κάποιο λάθος και να ξαναχρησιμοποιήσουν το έργο προσθέτοντας ένα ή περισσότερα νέα στοιχεία.

Και οι τρεις προσεγγίσεις είχαν δυνατά και λιγότερα δυνατά σημεία και σαν σύνολο η μια συμπλήρωνε και ολοκλήρωνε την προηγούμενη. Η αξιολόγηση μόνο της πολυπλοκότητας ενός έργου (1) δεν συνάδει την πλήρη εξοικείωση του μαθητή με τις έννοιες που χρησιμοποιεί και δεν μπορεί από μόνη της να αξιολογήσει την ανάπτυξη της υπολογιστικής σκέψης στον μαθητή, επομένως τα αποτελέσματα περιορίζονται στην διάσταση των Υπολογιστικών Εννοιών. Με τις συνεντεύξεις (2) ήταν πιο εύκολο να διακριθούν οι Υπολογιστικές Πρακτικές και Έννοιες που αναπτύσσει ο μαθητής μέσα από τα έργα του. Παρατηρήθηκε ότι ενώ ορισμένοι μαθητές είχαν δημιουργήσει έργα με μεγάλη πολυπλοκότητα (1) στις συνεντεύξεις προέκυψαν αρκετά εννοιολογικά κενά. Με την χρήση των σεναρίων σχεδιασμού (3) ήταν πιο εύκολο να παρατηρηθούν Υπολογιστικές Πρακτικές όπως η σταδιακή εξέλιξη ενός έργου, η εύρεση λαθών, η επαναχρησιμοποίηση και η τμηματοποιήση ενός έργου, στην πράξη. Στις συνεντεύξεις (2) οι μαθητές καλούνταν να ανακαλέσουν περιστάσεις στις οποίες είχαν χρησιμοποιήσει τις παραπάνω τεχνικές αλλά αυτό περιοριζόταν μόνο στην μνήμη των μαθητών και στην ικανότητας του να συνδέσουν

τη δράση τους με τις πρακτικές αυτές, στην πράξη (3) ωστόσο ο παρατηρητής (ο καθηγητής ή αυτός που κάνει την αξιολόγηση) είναι πιο εύκολο να διακρίνει πως ο μαθητής χειρίζεται αυτές τις πρακτικές και την λογική πίσω από την εφαρμογή τους. Το μειονέκτημα της τρίτης προσέγγισης είναι ότι η επιλογή των έργων έγινε από τρίτους και όχι από τους ίδιους τους μαθητές. Η διάσταση των Υπολογιστικών Αντιλήψεων ήταν η λιγότερο εμφανής στην έρευνά τους καθώς στην δεύτερη και τρίτη προσέγγιση εμφανίστηκαν στοιχεία αυτής της διάστασης στους μαθητές αλλά ήταν δύσκολο να ερωτηθούν ευθέως για τις αντιλήψεις που έχουν αναπτύξει.

2.4.3. Το περιβάλλον Kodu και η χρήση της Μάθησης με Υποστήριξη (scaffolding) στην ανάπτυξη ΥΣ

Με τον όρο Μάθηση με Υποστήριξη αναφερόμαστε στην παροχή βοήθειας στους μαθητές κατά την διάρκεια επίλυσης ενός προβλήματος. Σύμφωνα με την έρευνα των Lye & Koh, 2014 η πιο δημοφιλής προσέγγιση για την ανάπτυξη της ΥΣ σε μαθητές είναι η κατασκευή των δικών τους προγραμμάτων με την χρήση υποβοήθειας (scaffodls). Ο ρόλος την μάθησης με υποστήριξη μπορεί να είναι είτε μέσω εργαλείων υποβοήθειας ενσωματομένων στο προγραμματιστικό περιβάλλον που θα χρησιμοποιηθεί είτε μέσω οδηγιών που θα δίνονται στους μαθητές από τον διδάσκοντα/εκπαιδευτικό.

Σύμφωνα με τους Wood, Bruner, & Ross, 1976 ο ρόλος που έχει ο εκπαιδευτικός στην επίλυση προβλημάτων με την χρήση της μάθησης με υποστήριξη είναι

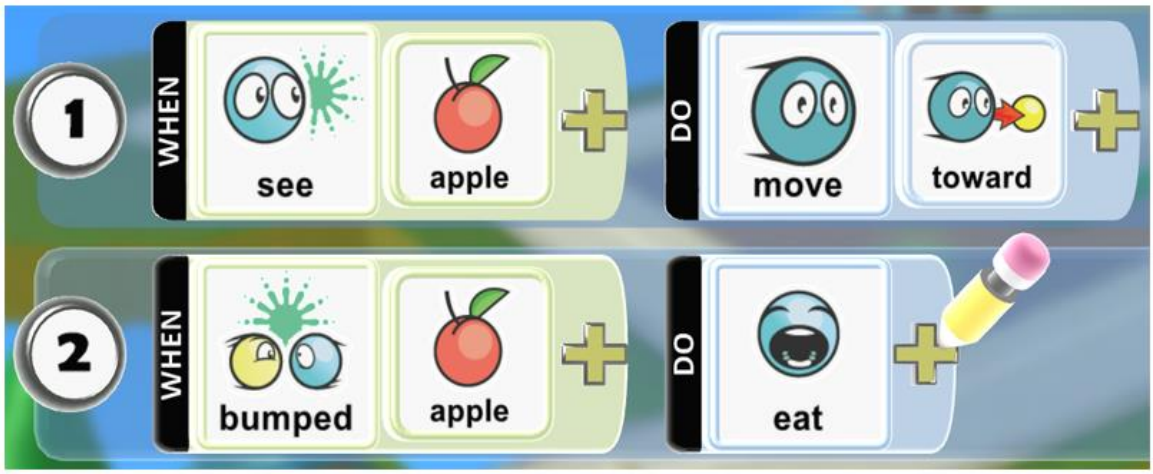
- να διατηρήσουν το ενδιαφέρον των μαθητών στο συγκεκριμένο πρόβλημα
- να κάνουν το πρόβλημα επιλύσιμο μειώνοντας τον βαθμό ελευθερίας των μαθητών
- να επισημαίνουν χαρακτηριστικά που μπορούν να βοηθήσουν τους μαθητές στην ολοκλήρωση του προβλήματος
- να δίνουν κίνητρα και να παρέχουν έγκαιρη καθοδήγηση ώστε να ελέγξουν την απογοήτευση των μαθητών και την εγκατάλειψη του προβλήματος από αυτούς
- να παρουσιάζουν ένα πρότυπο μοντέλο λύσης που να ενσωματώνει τις διαδικασίες που είναι απαραίτητες για την επίλυση του προβλήματος (προσομοίωση λύσης)

Οι παραπάνω πρακτικές προτείνονται (Lye & Koh, 2014) ως πρότυπο για την χρήση της μάθησης με υποστήριξη στην ανάπτυξη της υπολογιστικής σκέψης μέσω προγραμματισμού.

Όσον αφορά τα εργαλεία υποβοήθειας (scaffolding features) είναι έξυπνοι μηχανισμοί που διαθέτουν ορισμένα περιβάλλοντα ανάπτυξης ΥΣ όπως το Kodu που βοηθούν τους αρχάριους χρήστες στην αποφυγή λαθών (Touretzky, Marghitu, Ludi, Bernstein, & Ni,

2013). Όπως είδαμε στην παράγραφο 2.3.1 το Kodu Game Lab (http://www.kodugamelab.com/) ή απλά Kodu είναι ένα εργαλείο οπτικού προγραμματισμού για τη δημιουργία 3D παιχνιδιών. Ενσωματώνει μια γλώσσα οπτικού προγραμματισμού και με τα εργαλεία της ο χρήστης μπορεί να δημιουργήσει του επιθυμητούς κόσμους του και να προσθέσει χαρακτήρες(Εικόνα 8), ενώ με λίγες γραμμές κώδικα, με ακολουθίες εντολών τύπου WHEN/DO και με επιλογή γεγονότων-δράσεων μπορεί να προγραμματίσει τις κινήσεις του χαρακτήρα. Στην παρακάτω εικόνα βλέπουμε την χρήση της γλώσσας KGL από τον χρήστη για να προγραμματίσει τον χαρακτήρα του να βρει και να φάει όλα τα μήλα που έχουν απομείνει στον κόσμο που δημιουργήθηκε από τον ίδιο. Στην γραμμή ένα αυτού του κώδικα οπτικού προγραμματισμού με την εντολή WHEN όταν ο χαρακτήρας δει (γεγονός: see) ένα μήλο (αντικείμενο), θα προχωρήσει (δράση: move) προς το μήλο με την εντολή DO. Στην γραμμή 2 όταν ο χρήστης πέσει πάνω στο μήλο (γεγονός: bumped), συνθήκη που ελέγχεται και πάλι με την εντολή WHEN, θα το φάει με την εντολή DO και την δράση eat.

Εικόνα 7: Κώδικας σε KGL (πηγή: http://www.kodugamelab.com/)



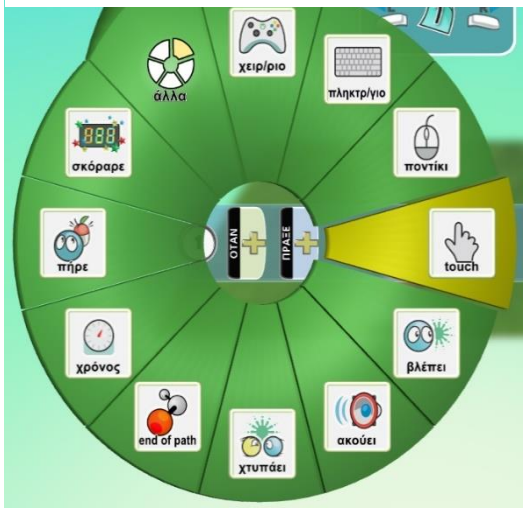
Η χρήση των εντολών WHEN/DO, σε συνδυασμό με την επιλογή εικονιδίων (γεγονότα - δράσεις) από ένα μενού συγκεκριμένων επιλογών για κάθε εντολή (Εικόνα 9), μηδενίζει την πιθανότητα συντακτικών λαθών στον κώδικα, επομένως το παιχνίδι παρέχει έντονη υποβοήθεια (Touretzky, Marghitu, Ludi, Bernstein, & Ni, 2013). Στο Kodu συναντάμε

- **Αντικείμενα** με την μορφή χαρακτήρων.
- **Μεταβλητές** με την μορφή score και χρονομέτρου που μπορεί να θέσει ο χρήστης για παράδειγμα όταν ολοκληρώνεται μια δράση ή για την πραγματοποίηση μιας δράσης μετά από κάποιο χρόνο.
- **Συνθήκες** με την χρήση των εντολών WHEN/DO.

- **Παραλληλισμό** με την ταυτόχρονη εκτέλεση πολλών εντολών στην ίδια σελίδα κώδικα καθώς στο Kodu μπορείς να έχεις πολλές σελίδες με κώδικα όπως της Εικόνα 7.
- **Βρόγχους επανάληψης** με έμμεσο τρόπο δεδομένου ότι μια ακολουθία εντολών θα εκτελείται επαναληπτικά εκτός αν διακοπεί από μια άλλη ακολουθία εντολών που θα εκτελεστεί μια φορά (με την χρήση του once) ή από αλλαγή σελίδας εντολών (υπάρχει το εικονίδιο switch). Επιπλέον συναντάμε και σαφείς βρόγχους επανάληψης με αρχικοποίηση, σώμα εντολών που θα επαναληφθούν με συνθήκη εξόδου (π.χ. WHEN…DO switch page 2) και εντολές τερματισμού, δηλαδή το σώμα των εντολών που θα εκτελεστεί μετά την επανάληψη. Συνήθως οι τρεις καταστάσεις του βρόγχου βρίσκονται σε διαφορετικές σελίδες.
- **Καταστάσεις** με την έννοια ότι κάθε σελίδα είναι μια κατάσταση για τον χαρακτήρα και η αλλαγή σελίδας σηματοδοτεί την αλλαγή κατάστασης.

Εικόνα 8: Επιλογές Kodu για την επεξεργασία του κόσμου και την δημιουργία χαρακτήρα

Εικόνα 9: Επιλογές που συνδέονται με τις εντολές WHEN (αριστερά) και DO (δεξιά)



Οι Touretzky, Marghitu, Ludi, Bernstein, & Ni δημοσιεύσαν το 2013 μια μελέτη που χρησιμοποιεί τα περιβάλλοντα Kodu, Alice και Lego Mindstorms NXT-G για την δημιουργία ενός μοντέλου (3 σταδίων), που αφορά την διδασκαλία προγραμματισμού και την ανάπτυξη της ΥΣ των μαθητών. Το μοντέλο διδασκαλίας τους αφορά την μετάβαση του μαθητή από ένα περιβάλλον που προσφέρει μεγάλο βαθμό υποβοήθησης (Kodu), σε ένα λιγότερο υποστηριζόμενο περιβάλλον (Alice) και τέλος σε ένα περιβάλλον χωρίς καθόλου υποβοήθηση (Lego NXT-G). Για την εφαρμογή του μοντέλου οργάνωσαν μια πενθήμερη καλοκαιρινή εκδήλωση στο πανεπιστήμιο του Auburn, στην οποία συμμετείχαν 31 μαθητές με μέσο όρο ηλικίας τα 13 έτη και εύρος ηλικιών 10-17. Οι πρώτες 2 μέρες της εκδήλωσης αφορούσαν τον προγραμματισμό με Kodu, οι επόμενες δυο με το πλαίσιο Alice και η τελευταία μέρα με το Lego NXT-G, που αφορά την ρομποτική, ενώ μαθητές εργάστηκαν σε ομάδες 3-5 ατόμων, με έναν ή δυο επιβλέποντες για 7 ώρες κάθε μέρα. Από την εφαρμογή του μοντέλου προέκυψε ότι η μετάβαση αυτή είναι εφικτή και ωφέλιμη για τους μαθητές. Κατά τη διάρκεια του έργου προέκυψαν αρκετές στρατηγικές για την διδασκαλία προγραμματισμού χρησιμοποιώντας το Kodu, αναφέρουμε μερικές από αυτές:

- Χρήση προγραμμάτων σε έτοιμους κόσμους με τοποθετημένους χαρακτήρες και αντικείμενα.
- Εμφάνιση του κώδικα του μαθητή σε κείμενο και έπειτα παρουσίαση του στο περιβάλλον Kodu με τη χρήση του οπτικού προγραμματισμού για να κατανοήσουν οι μαθητές ότι η ίδια ιδέα μπορεί να εκφραστεί
- Χρήση παρενθέσεων, που θα περιέχουν το όνομα του μενού/υπομενού ενός αντικειμένου όταν παρουσιάζεται ο κώδικας με την μορφή κειμένου. Σχεδόν κάθε επιλογή των μενού επιλογών του Kodu (Εικόνα 9) οδηγεί σε άλλο μενού με κίνδυνο να μη γνωρίζει ο μαθητής που βρίσκεται το κάθε αντικείμενο.
- Διδασκαλία του μηχανισμού καταστάσεων γιατί σαφή αναγνώριση των καταστάσεων με σελίδες.
- Διαχωρίσμός άμεσων και έμμεσων βρόγχων επανάληψης και χρήση διαγραμμάτων για την αποσαφήνισή τους.

2.4.4. Run Marco!

Το Run Marco είναι ένα εκπαιδευτικό παιχνίδι που έχει σαν στόχο να διδάξει στους χρήστες του βασικές έννοιες του προγραμματισμού όπως οι ακολουθίες, οι επαναλήψεις και οι συνθήκες (Giannakoulas & Xinogalos, 2020). Το παιχνίδι είναι μεταφρασμένο σε 26 γλώσσες, χρησιμοποιεί την βιβλιοθήκη της Blockly και είναι εξολοκλήρου βασισμένο στον ιστό. Αποτελείται από 36 επίπεδα διαβαθμισμένη δυσκολίας και ο χρήστης έχει την δυνατότητα να επιλέξει τον χαρακτήρα του ανάμεσα στην Σοφία και τον Μάρκο.

Στο σενάριο του παιχνιδιού ο χαρακτήρας του χρήστη έχει χαθεί στο δάσος και χρειάζεται βοήθεια για να βρει τους φίλους του. Στόχος του χρήστη σε κάθε επίπεδο είναι να οδηγήσει τον χαρακτήρα του μέσα από ένα μονοπάτι στο τέλος της διαδρομής, που είναι επισημασμένη με ένα κίτρινο αστέρι, έχοντας στην διάθεση του μια εργαλειοθήκη με μπλοκ. Η εργαλειοθήκη στα πρώτα επίπεδα περιέχει απλά μπλοκ τύπου προχώρησε μπροστά και κατά την εξέλιξη του παιχνιδιού είναι διαθέσιμα στον χρήστη και πιο σύνθετα μπλοκ όπως τα επανέλαβε κ φόρες, εάν συνθήκη κάνε κτλ. (Εικόνα 11) .

Εικόνα 10: Το παιχνίδι Run Marco!

Εικόνα 11: Διαθέσιμα μπλοκ στο Run Marco!

Για τον εντοπισμό λαθών το παιχνίδι παρέχει ένα πλήκτρο τύπου Run για την εκτέλεση του κώδικα και για να μεταβεί ο παίκτης σε επόμενο επίπεδο πρέπει ο κώδικάς του να είναι σωστός, ενώ από το μενού επιλογών (Εικόνα 12) έχει την δυνατότητα να επιλέξει το βελάκι πίσω που παίζει το ρόλο της επανεκκίνησης του επιπέδου. Ο σωστός κώδικας

βαθμολογείται με 1,2 ή 3 αστέρια ανάλογα με την αποδοτικότητα της λύσης του χρήστη, ενώ ο τελευταίος έχει τη δυνατότητα πλοήγησης στον χάρτη με τα επίπεδα και τα αστέρια που αντιστοιχούν στο κάθε επίπεδο επιλέγοντας τον χάρτη από το μενού επίλογών. Τα δεδομένα του επιπέδου που έχουν ολοκληρωθεί σωστά και η βαθμολογία τους σώζονται τοπικά στον φυλλομετρητή.

Εικόνα 12: Βοηθητικό μενού επιλογών του Run Marco!

Σε περίπτωση που η προσπάθεια του χρήστη είναι αποτυχημένη δεν υπάρχει ανατροφοδότηση για το λάθος του, ωστόσο κάθε φορά που εισάγεται ένα μπλοκ με πιο σύνθετη λειτουργικότητα σε κάποιο επίπεδο, το παιχνίδι δίνει για αυτό πληροφορίες στον χρήστη και τον καθοδηγεί με ένα βελάκι στα σωστά μπλοκ που πρέπει να επιλεγούν για να επιλύσει το πρόβλημα του επιπέδου, ενώ στο επόμενο επίπεδο ο χρήστης θα συναντήσει παρόμοιο πρόβλημα για να εμπεδώσει την χρήση των νέων μπλοκ.

Τα επίπεδα του παιχνιδιού μπορούν να χωριστούν σε 6 ενότητες, με βάση τα νέα μπλοκ και τις λειτουργικότητες που εισάγονται (Giannakoulas & Xinogalos, 2018). Στον παρακάτω πίνακα συνοψίζονται οι κατηγορίες αυτές.

Πίνακας 4: Οι 6 ενότητες των επιπέδων του Run Marco
(πηγή: Giannakoulas & Xinogalos, 2018)

Ενότητα		Περιεχόμενο	Επίπεδα
1	Ακολουθία Εντολών	Ακολουθία εντολών εκτέλεσης	1-9
2	Επανάληψη I	Βρόγχοι επανάληψης με προκαθορισμένο αριθμό επαναλήψεων και φωλιασμένοι βρόγχοι.	10-19
3	Επανάληψη II	Βρόγχοι με απροσδιόριστο αριθμό επαναλήψεων	20-22
4	Συνθήκες I	Απλό if	23-30
5	Συνθήκες II	If…else	31-33
6	Συνθήκες III	Φωλιασμένα If	34-36

Στην έρευνα που δημοσίευσαν οι Γιαννακούλας και Ξυνόγαλος το 2018 επέλεξαν το παιχνίδι Run Marco για την εισαγωγή νεαρών μαθητών στις έννοιες του προγραμματισμού και της υπολογιστικής σκέψης. Τα υπόλοιπα παιχνίδια που μελετήθηκαν στα πλαίσια της έρευνάς τους με παρόμοια χαρακτηριστικά ήταν τα CodeMonkey, Getgoding, Kodable, Lightbot (1.0 & 2.0), Program Your Robot και RapidRouter.

Στα πλαίσια της έρευνας δημιουργήθηκαν 6 σχέδια μαθημάτων ένα για την κάθε ενότητα του παιχνιδιού (Πίνακας 4), ωστόσο για την διεξαγωγή της έρευνας χρησιμοποιήθηκαν μόνο τρία σχέδια μαθημάτων που αφορούν τις 3 πρώτες ενότητες του παιχνιδιού. Η έρευνα έλαβε χώρα σε ένα δημοτικό σχολείο της Θεσσαλονίκης στα πλαίσια του μαθήματος «Τεχνολογίες Πληροφορίας και Επικοινωνιών (ΤΠΕ)», είχε συνολική διάρκεια 3 ώρες και συμμετείχαν σε αυτή 20 μαθητές της Ε΄ τάξης. Κάθε σχέδιο μαθήματος είχε τα εξής χαρακτηριστικά:

- Είχε διάρκεια μιας ώρας.
- Αρχικά δόθηκε στους μαθητές ένα φύλλο δραστηριοτήτων με οδηγίες για τα επίπεδα της ενότητας του μαθήματος.
- Οι μαθητές χωρίστηκαν σε ζευγάρια και είχαν στην διάθεσή τους συγκεκριμένο χρόνο για να παίξουν τα επίπεδα της ενότητας στο περιβάλλον του Run Marco.
- Με την ολοκλήρωση των επιπέδων κάθε μαθητής συμπλήρωνε γραπτά, ένα φύλλο αξιολόγησης με ασκήσεις και ερωτήσεις σχετικές με τις έννοιες που διαπραγματεύτηκε στην ενότητα αυτή.

Τέλος, με την ολοκλήρωση των 3 μαθημάτων οι μαθητές συμπλήρωσαν ένα ερωτηματολόγιο που αφορά την Αντιλαμβανόμενη Χρησιμότητα, την Ευκολία Χρήσης και την Πρόθεση Χρήσης στο μέλλον του παιχνιδιού Run Marco, το οποίο σχεδιάστηκε σύμφωνα με το μοντέλο Technology Acceptance Model. Από την έρευνα προέκυψαν τα εξής συμπεράσματα (Giannakoulas & Xinogalos, 2018) για τις ενότητες του παιχνιδι:

- Το φύλλο δραστηριοτήτων, σύμφωνα με τις παρατηρήσεις του καθηγητή, ήταν αρκετά βοηθητικό στους μαθητές.
- Οι μαθητές δεν συνάντησαν ιδιαίτερες δυσκολίες στην πρώτη ενότητα, στην δεύτερη ενότητα πάνω από το 60% των μαθητών απάντησε σωστά όλες τις ερωτήσεις του φύλλου αξιολόγησης, ενώ στην τρίτη ενότητα η πλειοψηφία των μαθητών απάντησε σωστά στις 3 από τις 4 ερωτήσεις του φύλλου αξιολόγησης.
- Αναφορικά με το ερωτηματολόγιο που συμπλήρωσαν οι μαθητές για την αξιολόγηση του Run Marco το μεγαλύτερο μέρος των μαθητών βρήκε τους κανόνες του παιχνιδιού κατανοητούς, δήλωσε ότι δεν χρειάστηκε βοήθεια στην ολοκλήρωση των επιπέδων και μπορούσε εύκολα να εντοπίσει τα λάθη στο πρόγραμμά του. Ωστόσο μόνο το 52,6% έμεινε ευχαριστημένο από την συχνότητα των μηνυμάτων βοήθειας στο παιχνίδι. Τέλος περίπου οι μισοί

μαθητές είχαν θετική στάση απέναντι στο παιχνίδι και στην χρήση του στο μάθημα ΤΠΕ, ενώ το 68.42% συμφώνησε ότι το παιχνίδι τους έδωσε την ευκαιρία να συμμετέχουν πιο άνετα στο μάθημα και το 89.48% θεωρεί ότι το παιχνίδι τους βοήθησε στην κατανόηση των προγραμματιστικών εννοιών που παρουσιάστηκαν.

2.4.5. Σύντομη παρουσίαση των Blockly Games

Τα Παιχνίδια Blockly είναι μια σειρά εκπαιδευτικών παιχνιδιών για την εκμάθηση προγραμματισμού μέσω ενός κατευθυνόμενου σεναρίου αυξανόμενης δυσκολίας. Ο πηγαίος κώδικας των Παιχνιδιών Blockly χρησιμοποιεί την βιβλιοθήκη Blockly και την γλώσσα προγραμματισμού JavaScript. Ο κώδικας των Blockly Games είναι ανοιχτός και διαθέσιμος στο GitHub, στο Παράρτημα Β της εργασίας παρουσιάζονται τα βήματα για την ανάπτυξη των παιχνιδιών σε τοπικό περιβάλλον.

Τα παιχνίδια Blockly είναι μια προσπάθεια εισαγωγής της βιβλιοθήκης της Blockly σε εκπαιδευτικά παιχνίδια απευθύνονται κυρίως σε μαθητές και αρχάριους που θέλουν να κάνουν τα πρώτα τους βήματα στον προγραμματισμό, αντιμετωπίζοντας μόνο την λογική του κώδικα και όχι την συντακτική συνέπειά του.

Εικόνα 13: Λογότυπο Blockly Games

Η σειρά των παιχνιδιών Blockly αποτελείται από 7 παιχνίδια. Κάθε παιχνίδι έχει 10 επίπεδα αυξανόμενης δυσκολίας και στο τέλος κάθε παιχνιδιού παρουσιάζεται ο κώδικας των μπλοκ που χρησιμοποίησε ο χρήστης για να ολοκληρώσει το ζητούμενο του επιπέδου, η παρουσίαση του κώδικα γίνεται σε γλώσσα JavaScript (Εικόνα 14).

Η διεπαφή των παιχνιδιών αποτελείται από ένα πάνελ στο οποίο βρίσκονται τα γραφικά του παιχνιδιού και από μια κλασική διεπαφή του συντάκτη της Blockly. Τα μπλοκ που παρουσιάζονται σε κάθε παιχνίδι έχουν διαφορετικές λειτουργικότητες.

Πίνακας 5: Παιχνίδια Blockly

Λογότυπο	Όνομα και περιγραφή παιχνιδιού
	**Παιχνίδι 1: Παζλ (Puzzle)** Είναι το εισαγωγικό παιχνίδι με το οποίο ο χρήστης μαθαίνει να ενώνει τα μπλοκ για να δημιουργεί κώδικα.
	**Παιχνίδι 2: Λαβύρινθος (Maze)** Στο παιχνίδι αυτό ο χρήστης χρησιμοποιεί τα μπλοκ για να οδηγήσει το εικονίδιο του παίκτη στο τέλος του λαβυρίνθου. Ξεκινά με έναν απλό λαβύρινθο που εισάγει στον χρήστη τις έννοιες προχώρησε μπροστά και στρίψε. Σε κάθε επίπεδο εισάγονται νέα μπλοκ που εξοικειώνουν τον χρήστη με τη λογική της συνθήκης και του βρόχου. Επιπλέον από ένα επίπεδο και μετά ο χρήστης έχει περιορισμένο αριθμό μπλοκ που μπορεί να χρησιμοποιήσει.
	**Παιχνίδι 3: Πουλί (Bird)** Ο χρήστης χρησιμοποιεί τα μπλοκ για να ελέγξει την πορεία του πουλιού και να πάει το σκουλήκι στη φωλιά, καθώς ανεβαίνουν τα επίπεδα οι συνθήκες γίνονται πιο πολύπλοκες. Σε αυτό το παιχνίδι εισάγεται η κίνηση υπό γωνία.
	**Παιχνίδι 4: Χελώνα (Turtle)** Σε αυτό το παιχνίδι ο χρησιμοποιεί κώδικα για να σχεδιάσει τα ζητούμενα σχήματα. Χρησιμοποιούνται εντολές όπως κινήσου μπροστά 100px (pixel), κίνηση και στροφή υπό γωνία και αλλαγή χρώματος. Ο χρήστης μπορεί να χρησιμοποιήσει εμφωλευμένους βρόγχους για να επιτύχει το ζητούμενο.
	**Παιχνίδι 5: Ταινία (Movie)** Χρησιμοποιούνται μαθηματικοί τύποι για να χρωματιστούν τα διάφορα σχήματα και καθώς ανεβαίνουν τα επίπεδα να δημιουργηθεί μια ταινία από τον χρήστη. Ο κώδικας των μπλοκ σε αυτό το παιχνίδι μπορεί να γίνει αρκετά πολύπλοκος σε μεγαλύτερα επίπεδα.
	**Παιχνίδι 6: Μουσική (Music)** Με το παιχνίδι μουσική στόχος του χρήστη είναι να παίξει ένα μουσικό κομμάτι χρησιμοποιώντας τα μπλοκ. Τα μπλοκ δίνουν τη δυνατότητα να παίξει συγκεκριμένες νότες από διάφορα μουσικά όργανα. Ο κώδικας που πρέπει να σχηματιστεί με τα μπλοκ γίνεται

αρκετά δύσκολος προς τα μεγαλύτερα επίπεδα και ο χρήστης πρέπει να μπορεί να εκτελεί παράλληλα μπλοκ.

Παιχνίδι 7: Εκπαιδευτής Pond (Pond-tutor)

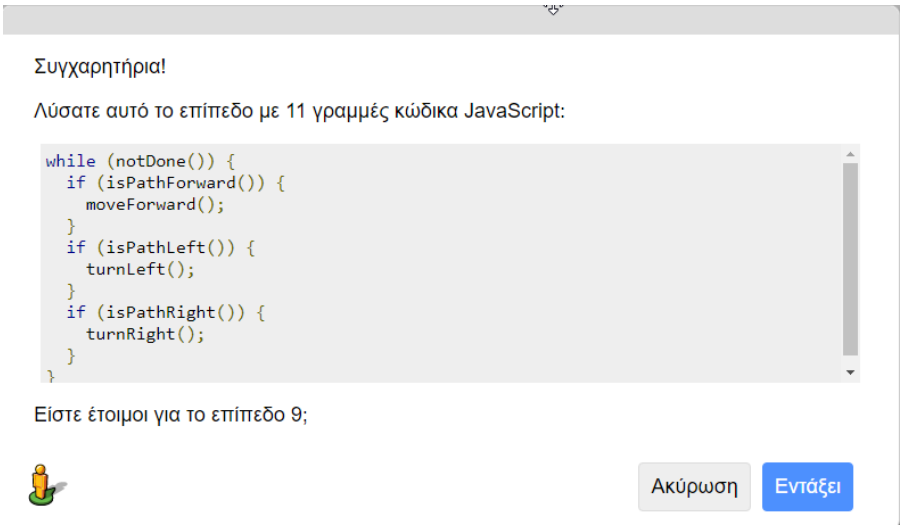
Σε αυτό και το επόμενο παιχνίδι γίνεται η μετάβαση από τον κώδικα με μπλοκ σε κλασικό κώδικα κειμένου (text-based). Ο χρήστης από το δεύτερο επίπεδο και μετά γράφει τον κώδικα που πρέπει να εκτελεστεί για να χτυπήσει το στόχο. Στο παιχνίδι αυτό παρέχεται και η τεκμηρίωση του, που δείχνει στο χρήστη τις συναρτήσεις, τα ορίσματα, τους βρόχους και τις συνθήκες.

Παιχνίδι 8: Λίμνη (Pond-duck)

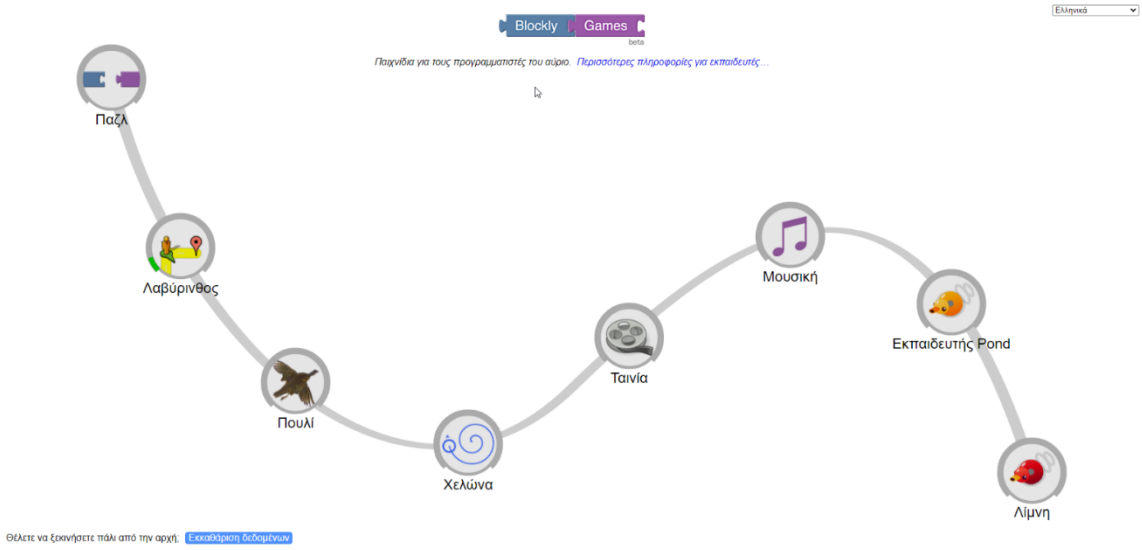
Το τελευταίο παιχνίδι δεν έχει επίπεδα. Ο χρήστης πρέπει να προγραμματίσει την πιο έξυπνη πάπια χρησιμοποιώντας μπλοκ ή JavaScript. Είναι το πιο διαδραστικό από όλα τα παιχνίδια και παρέχει και αυτό τεκμηρίωση στο χρήστη. Ο χρήστης μπορεί να δει τον κώδικα JavaScript με τον οποίο κινούνται όλες οι πάπιες και να προγραμματίσει την δική του αναλόγως.

Στην αρχική σελίδα των Παιχνιδιών Blockly ο χρήστης μπορεί να δει την πορεία του σε κάθε παιχνίδι και να κάνει εκκαθάριση δεδομένων εφόσον το επιθυμεί. Τα Παιχνίδια Blockly έχουν τη δυνατότητα να κρατάνε τα αρχεία των ολοκληρωμένων επιπέδων και κάθε φορά που επισκέπτεται ο χρήστης στην ιστοσελίδα των παιχνιδιών από τον ίδιο φυλλομετρητή μπορεί να συνεχίζει το παιχνίδι από το σημείο που το είχε σταματήσει. Τέλος ο χρήστης έχει τη δυνατότητα να επιλέξει τη γλώσσα που επιθυμεί και σε ορισμένα παιχνίδια μπορεί να αλλάξει τον χαρακτήρα του παίκτη και το φόντο του παιχνιδιού χωρίς να χαθούν τα δεδομένα του.

Εικόνα 14: Μήνυμα επιτυχούς ολοκλήρωσης επιπέδου



Εικόνα 15: Αρχική σελίδα Παιχνιδιών Blockly



Ανάλυση των Παιχνιδιών Blockly

Τα Παιχνίδια Blockly ανήκουν στην κατηγορία Παιχνίδι πρόκλησης. Τα χαρακτηριστικά αυτού του είδους είναι ότι αποτελούνται από μια σειρά προγραμματιστικών δοκιμασιών με την μορφή προσχεδιασμένων επιπέδων για επίλυση συνήθως εμφανίζοντας νέα προγραμματιστικά εργαλεία σε κάθε επίπεδο και διαβαθμισμένη δυσκολία. Στον παρακάτω πίνακα παρουσιάζονται τα βασικά χαρακτηριστικά των Blockly Games.

Πίνακας 6: Ανάλυση των παιχνιδιών Blockly (Eguiluz, Garaizar, & Guenaga, 2018)

Γενικά Χαρακτηριστικά	Τύπος Παιχνιδιού	Παιχνίδι πρόκλησης (challenge)
	Χώρα Προέλευσης	ΗΠΑ
	Έτος	2014
	Υποστηριζόμενες γλώσσες	49
	Τεχνολογία	Εφαρμογή ιστού
Προτεινόμενες Ηλικίες χρήσης παιχνιδιού	<5	Δεν συστήνεται
	6-7	Δεν συστήνεται
	8-9	Συστήνεται
	10-11	Συστήνεται
	12-13	Συστήνεται
	14-15	Συστήνεται
	16-17	Βιώσιμο
	>18	Βιώσιμο
Άλλες εκπαιδευτικές διαστάσεις και απλότητα εγκατάστασης	Εγκατάσταση[3]	0
	Προσωπικά δεδομένα που ζητούνται	Κανένα
	Χρηστικότητα[3]	0
	Πλούτος Αλληλεπίδρασης[6]	0
	Χρόνος αφοσίωσης εκπαιδευόμενου	1ώρα – 15μέρες
	Χρόνος προετοιμασίας εκπαιδευτικού	5ώρες – 20ώρες

[6] Αξιολογείται στο εύρος 0-3, όπου 0 είναι το πιο απλό και 3 το πιο σύνθετο

	Διαθέσιμο περιεχόμενο για εκπαιδευτικούς	Πλούσιο περιεχόμενο σε πολλές γλώσσες.
Αφοσίωση του χρήστη συσχετισμένη με τις διαφορετικές διαστάσεις της διασκέδασης που παρέχονται από το παιχνίδι[7]	Αίσθηση	1
	Φαντασία	1
	Αφήγηση	1
	Προκλήσεις	3
	Συντροφικότητα	1
	Εξερεύνηση	2
	Έκφραση	2
	Υποβολή	2
	Διήγηση	2
Στοιχεία Υπολογιστικής σκέψης	Βρόχοι	ΝΑΙ
	Εναλλακτικές δομές τύπου Αν-Αλλιώς	ΝΑΙ
	Οπτική Αποσφαλμάτωση στην εκτέλεση του κώδικα	ΝΑΙ
	Ενότητες (Υποπρογράμματα)	ΝΑΙ
	Μεταβλητές	ΝΑΙ
	Εκφράσεις	ΝΑΙ
	Αριθμός μπλοκ για την κατασκευή κώδικα	136
	Συμβάντα	ΟΧΙ
	Πολλαπλά νήματα	ΟΧΙ
	Αναδρομή	ΟΧΙ
	Μηνύματα	ΟΧΙ
	Αντίστοιχη γλώσσα κειμένου	ΝΑΙ
	Γλώσσα	JavaScript
Σχεδιαστικές θεωρήσεις	Ολοκληρωμένες Οδηγίες	ΝΑΙ
	Ολοκληρωμένη βοήθεια	ΝΑΙ

[7] ελάχιστο =1 , μέγιστο=3

	Δωρεάν Πλοήγηση	ΝΑΙ
	Απαραίτητη η σύνδεση χρήστη	ΟΧΙ
	Δημιουργία Ομάδας	ΟΧΙ
	Ανατροφοδότηση σχολίων για την συμπεριφορά του χρήστη	ΟΧΙ
	Ταμπλό εκπαιδευτή	ΟΧΙ
	Δημόσια χρήση δεδομένων χρήστη	ΟΧΙ
	Δημόσια αναζήτηση στα δεδομένα του χρήστη	ΟΧΙ
Ανατροφοδότηση χρήστη	Ανατροφοδότηση πριν από κάθε επίπεδο	ΝΑΙ
	Ανατροφοδότηση μετά από αποτυχημένη προσπάθεια	ΟΧΙ
	Ανατροφοδότηση μετά από επιτυχημένη ολοκλήρωση επιπέδου	ΝΑΙ
	Πληροφορίες για την πορεία του χρήστη στο παιχνίδι	ΝΑΙ (πληροφορίες για τα επίπεδα που έχουν ολοκληρωθεί)

2.5. Συμπεράσματα

Σε αυτό το κεφάλαιο παρουσιάστηκαν οι έννοιες της Υπολογιστικής Σκέψης, της Μάθησης με Υποστήριξης και των περιβαλλόντων οπτικού προγραμματισμού. Παρουσιάσαμε μερικές από τις πιο δημοφιλείς γλώσσες οπτικού προγραμματισμού και αναλύσαμε 2 περιβάλλοντα οπτικού προγραμματισμού που χρησιμοποιήθηκαν ως πλαίσια ανάπτυξης της ΥΣ και δυο εκπαιδευτικά παιχνίδια πρόκλησης που το ένα σχεδιάστηκε με βάση ενός πλαισίου ΥΣ, ενώ το δεύτερο χρησιμοποιήθηκε στην κατάρτιση ενός σχεδίου μαθήματος

για την εκμάθηση του προγραμματισμού σε μικρές ηλικίες μαθητών. Από την έρευνα που πραγματοποιήθηκε είναι αντιληπτή η ανάγκη δημιουργίας ενός εκπαιδευτικού εργαλείου για την αξιολόγηση της ΥΣ που θα χρησιμοποιεί παιδαγωγικές μεθόδους, όπως το πλαίσιο Brennan και Resnick και θα χρησιμοποιεί την μάθηση με υποστήριξη και εργαλεία υποβοήθειας. Για την ανάπτυξη αυτού του εργαλείου επιλέξαμε ως βάση τα παιχνίδια Blockly των οποίων τα χαρακτηριστικά αναλύουμε σε αυτό το κεφάλαιο.

3. Μεθοδολογία

3.1. Εισαγωγή

Στο κεφάλαιο αυτό παρατίθεται η μεθοδολογία που ακολουθήθηκε για την εκπόνηση της παρούσας διπλωματικής εργασίας. Περιγράφονται συνοπτικά τα βήματα που ακολουθήθηκαν από την αναζήτηση της βιβλιογραφίας και την εξοικείωση με τις απαραίτητες τεχνολογίες ως την σχεδίαση και την υλοποίηση του εκπαιδευτικού παιχνιδιού aMazeD. Το παιχνίδι aMazeD σχεδιάστηκε και αναπτύχθηκε με βάση το μοντέλο του καταρράκτη σε διακριτές φάσεις που συνοψίζονται στην αναζήτηση προτύπου, την οριοθέτηση των προδιαγραφών που πρέπει να πληρούνται, την μελέτη των απαραίτητων τεχνολογιών που πρόκειται να χρησιμοποιηθούν, την σχεδίαση και υλοποίηση του συστήματος με βάση τις προδιαγραφές.

3.2. Αναζήτηση και συλλογή πληροφοριών για την υπολογιστική σκέψη και την βιβλιοθήκη της Blockly

Το πρώτο στάδιο της εργασίας ήταν κυρίως βιβλιογραφική έρευνα και εξοικείωση με τις έννοιες που αφορούν την ανάπτυξη ενός περιβάλλοντος εκμάθησης υπολογιστικής σκέψης. Αρχικά μελετήθηκαν οι έννοιες της υπολογιστικής σκέψης, του οπτικού προγραμματισμού και των παιχνιδιών σοβαρού σκοπού. Μελετήθηκαν και αξιολογήθηκαν εκπαιδευτικά site όπως το hour of code (https://hourofcode.com/gr) και εκπαιδευτικά παιχνίδια που αναπτύσσουν τις ικανότητες της ΥΣ. Επιπλέον μελετήθηκαν διάφορα πλαίσια ΥΣ τέθηκαν οι βασικές αρχές που πρέπει να ικανοποιούνται στην ανάπτυξη του περιβάλλοντος υπολογιστικής σκέψης. Τέλος έγινε μια πρώτη επισκόπηση της βιβλιοθήκης Blockly και των Παιχνιδιών Blockly.

3.3. Εξοικείωση με τις τεχνολογίες που θα χρησιμοποιηθούν

Στο δεύτερο στάδιο μελετήθηκαν οι τεχνολογίες και τα εργαλεία που θεωρήθηκαν απαραίτητα για την ανάπτυξη του παιχνιδιού και την ολοκλήρωση της διπλωματικής εργασίας. Επιλέχθηκε το δωρεάν λογισμικό Visual Studio Code ως περιβάλλον εργασίας και ανάπτυξης κώδικα. Ένα από τα χαρακτηριστικά της ΔΕ είναι η λήψη και τροποποίηση ανοιχτού κώδικα, διαθέσιμου στο GitHub επομένως δημιουργήθηκε λογαριασμός χρήστη και μελετήθηκε η χρήση του GitHub και οι δυνατότητες που παρέχει για fork/clone σε

διαθέσιμες εφαρμογές (repositories) καθώς και η αντίστοιχη εφαρμογή του στον υπολογιστή το λογισμικό Git το οποίο χρησιμοποιήθηκε σε όλη τη διάρκεια της εργασίας για την αποθήκευση των αλλαγών της εφαρμογής και την ενημέρωση του αντίστοιχου repository στο GitHub. Η ανάπτυξη της εφαρμογής κατά το μεγαλύτερο μέρος της αφορά τη γλώσσα προγραμματισμού JavaScript, έτσι αφιερώθηκε ένα μεγάλο κομμάτι στην μελέτη της συγκεκριμένης γλώσσας και την εξοικείωση με στοιχεία της όπως είναι τα events, οι κλάσεις και οι συναρτήσεις καθώς και συ σχετιζόμενες τεχνολογίες όπως είναι το Node.js και ο npm. Τέλος μελετήθηκε η γλώσσα σήμανσης HTML, κάποιες ιδιότητες CSS και η χρήση του Xampp και του Node.js για στήσιμο τοπικού εξυπηρετητή .

3.4. Μελέτη της βιβλιοθήκης Blockly

Στο στάδιο αυτό μελετήθηκε η βιβλιοθήκη της Blockly από την επίσημη ιστοσελίδα της για προγραμματιστές. Έγινε εκτενής έρευνα στο τι αφορά τι είναι η Blockly πως μπορεί να γίνει λήψη της βιβλιοθήκης και ενσωμάτωσής της σε εφαρμογές.
Η Blockly όπως έχει ήδη αναφερθεί παρέχει πολλές δυνατότητες και αφιερώθηκε χρόνος για την εξοικείωση με τον συντάκτη της Blockly, τη γενέτειρα κώδικα και την δημιουργία μπλοκ με την χρήση των παραδειγμάτων που υπάρχουν στους φακέλους Examples και Demo της βιβλιοθήκης (αν γίνει λήψη όλων των αρχείων της από το GitHub).

3.5. Μελέτη Παιχνιδιών Blockly και τροποποίηση κώδικα

Στο στάδιο αυτό ασχοληθήκαμε με τα παιχνίδια Blockly. Στην ιστοσελίδα των παιχνιδιών https://blockly.games/ μελετήθηκε η διεπαφή των παιχνιδιών , τα επίπεδα και η χρήση του συντάκτη της Blockly.
Σε επόμενη φάση έγινε λήψη του κώδικα των παιχνιδιών από το GitHub και μελετήθηκε ο τρόπος με τον οποίο μπορούμε να τρέξουμε τα παιχνίδια σε τοπικό εξυπηρετητή με την χρήση του Xampp καθώς και πως μπορούν να γίνουν αλλαγές στα παιχνίδια Λαβύρινθος και Χελώνα τα οποία αποτελούν την βάση του παιχνιδιού που αναπτύχθηκε για την διπλωματική. Για να πραγματοποιηθούν τα παραπάνω χρειάστηκε αρκετός χρόνος για μελέτη της γραμμής εντολών των Windows, την χρήση γραμμών εντολών ψευδοκώδικα Linux στη γραμμή εντολών Bash του Git ή την γραμμή εντολών του Visual Studio Code και τελικά η χρήση του υποσυστήματος Ubuntu για να μπορέσουμε να κάνουμε build στις αλλαγές των παιχνιδιών. Με την ολοκλήρωση των παραπάνω έγιναν και οι πρώτες τροποποιήσεις στη διεπαφή, τα γραφικά και τα επίπεδα του Λαβυρίνθου.

3.6. Σχεδίαση και υλοποίηση παιχνιδιού aMazeD

Το στάδιο αυτό αφορά όλες τις ενέργειες και τις αποφάσεις που λήφθηκαν στη σχεδίαση και την υλοποίηση του παιχνιδιού aMazeD. Το παιχνίδι aMazeD ικανοποιεί συγκεκριμένες προδιαγραφές, οι εργασίες του παιχνιδιού έγιναν σταδιακά τροποποιώντας τον αρχικό κώδικα των παιχνιδιών Λαβύρινθος και Χελώνα με στόχο την υλοποίηση όλων των προδιαγραφών.

Δημιουργήθηκαν αρχικά οι δυο εκδόσεις του παιχνιδιού , μπήκαν τα σχόλια ανάδρασης , δημιουργήθηκε χρονόμετρο. Προστέθηκαν τα κουμπιά Υποβολή και Παίξε για έλεγχο και υποβολή του κώδικα, αφαιρέθηκε η δυνατότητα μετάβασης του χρήστη σε επιθυμητό επίπεδο και τέλος δημιουργήθηκαν οι βοηθητικές σελίδες με τις πληροφορίες και τα αποτελέσματα.

3.7. Προσθήκη Βαθμολογίας και διατήρηση των logs του χρήστη

Στο τελευταίο στάδιο έγινε η προσθήκη του κώδικα για την διατήρηση των δεδομένων του χρήστη ανά επίπεδο και την εμφάνιση τους στη σελίδα αποτελεσμάτων. Δημιουργήθηκαν οι κατάλληλες συναρτήσεις και τα αντικείμενα για την αποθήκευση τοπικά με τη χρήση του local storage των δεδομένων

- χρόνος επιπέδου
- πλήθος παίξε επιπέδου
- αποτέλεσμα επιπέδου
- βαθμολογία επιπέδου

Δημιουργήθηκε ένας συγκεντρωτικός πίνακας στη σελίδα αποτελεσμάτων που εμφανίζει τα παραπάνω δεδομένα και έγινε έρευνα για τους τρόπους με τους οποίους θα μπορούσε ο χρήστης να αποθηκεύει τα αποτελέσματα του παιχνιδιού.

3.8. Περίληψη

Στο κεφάλαιο αυτό παρουσιάστηκε η βασική μεθοδολογία με την οποία προσεγγίστηκε η ανάπτυξη του παιχνιδιού aMazeD. Σύμφωνα με όσα αναφέρθηκαν η ανάπτυξη του παιχνιδιού στηρίχτηκε στην μελέτη της Blockly, της JavaScript, και της HTML καθώς και των τεχνολογιών Ubuntu, GitHub, Xampp και η χρήση των γνώσεων αυτών στην τροποποίηση και την εξέλιξη του κώδικα των παιχνιδιών Λαβύρινθος και Χελώνα, των

Παιχνιδιών Blockly, για την υλοποίηση του παιχνιδιού aMazeD σύμφωνα με τις σχεδιαστικές προδιαγραφές που τέθηκαν.

4. Περιγραφή του εκπαιδευτικού παιχνιδιού aMazeD

4.1. Εισαγωγή

Σε αυτό το κεφάλαιο παρουσιάζεται το παιχνίδι aMazeD που δημιουργήθηκε στα πλαίσια της παρούσας διπλωματικής εργασίας. Το aMazeD σχεδιάστηκε βάση των εργαλείων που αναπτύχθηκαν στο κεφάλαιο 2 . Αρχικά αναφέρουμε ποιοι είναι οι στόχοι της σχεδίασης του και τις προδιαγραφές που πρέπει να πληροί το παιχνίδι για την επίτευξή τους. Έπειτα παρουσιάζονται οι δυο εκδόσεις στις οποίες είναι διαθέσιμο το παιχνίδι, η διεπαφή του παιχνιδιού, τα γραφικά, τα επίπεδα, τα σχόλια βοήθειας, τα διαθέσιμα μπλοκ και οι λύσεις για κάθε επίπεδο. Τέλος παρουσιάζονται οι σελίδες πληροφοριών και αποτελεσμάτων που συνοδεύουν την κυρίως εφαρμογή του παιχνιδιού και αναλύονται τα χαρακτηριστικά του aMazeD αναφορικά με τους στόχους σχεδίασης του.

4.2. Προδιαγραφές παιχνιδιού

Η σχεδίαση και υλοποίηση του παιχνιδιού aMazeD έχει δυο βασικούς στόχους:

1) ο χρήστης πρέπει να χρησιμοποιήσει κάποιες από τις υπολογιστικές έννοιες και πρακτικές που συναντάμε στο πλαίσιο ΥΣ των Brennan και Resnick και
2) το παιχνίδι πρέπει να ενσωματώνει την Βοήθεια με Υποστήριξη.

Για να επιτευχθούν τα παραπάνω τροποποιήθηκε ο κώδικας των παιχνιδιών Λαβύρινθος(Maze) και Χελώνα (Turtle) που συμπεριλαμβάνονται στα παιχνίδια Blockly σύμφωνα με τις παρακάτω προδιαγραφές:

- Αλλαγή στα προβλήματα που αντιμετωπίζει ο χρήστης σε κάθε επίπεδο και στην πλοήγηση του σε επόμενο ή προηγούμενο επίπεδο.
- Αλλαγή στα γραφικά των επιπέδων.
- Αλλαγή στην εμφάνιση των μηνυμάτων βοήθειας σε κάθε επίπεδο τόσο λεκτικά όσο και στην λειτουργικότητα τους.
- Αλλαγή στα μπλοκ που εμφανίζονται στην εργαλειοθήκη του χρήστη και προσθήκη νέων μπλοκ όπου είναι απαραίτητο.
- Προσθήκη ενός κουμπιού «Παίξε» του οποίου η λειτουργικότητα θα είναι να φορτώνει τον κώδικα και να τον εκτελεί χωρίς να ελέγχει την ορθότητα του κώδικα.

- Προσθήκη ενός κουμπιού «Υποβολή» για την αξιολόγηση και την αποθήκευση του αποτελέσματος του επιπέδου και την μετάβαση στο επόμενο επίπεδο.
- Προσθήκη χρονομέτρου σε κάθε επίπεδο.
- Αποθήκευση και εμφάνιση επιπλέον πληροφοριών στο χρήστη για την καλύτερη αξιολόγηση της λύσης του όπως:
    - η μέτρηση των φορών που πατήθηκε το κουμπί «Παίξε»
    - ο χρόνος που χρειάστηκε για να επιλύσει το επίπεδο
    - το αποτέλεσμα του επιπέδου χαρακτηρισμένο ως Επιτυχία/Αποτυχία
    - η βαθμολογία του κώδικά του αν επιλύει σωστά το επίπεδο
    - ο κώδικας του χρήστη σε γλώσσα JavaScript είτε επιλύει το επίπεδο επιτυχημένα είτε όχι
- Δημιουργία μιας έκδοσης του παιχνιδιού που να εμφανίζει ένα μέρος της λύσης με την μορφή μπλοκ στον χώρο εργασίας του χρήστη, οδηγίες στον χρήστη για την συμπλήρωση του κώδικα και ανατροφοδότηση του μετά από λάθος λύση.
- Δημιουργία μιας δεύτερης έκδοσης του παιχνιδιού χωρίς προ συμπληρωμένο κώδικα και με ελάχιστη βοήθεια σε κάθε επίπεδο.

Εικόνα 16: Λογότυπο παιχνιδιού aMazeD[8]

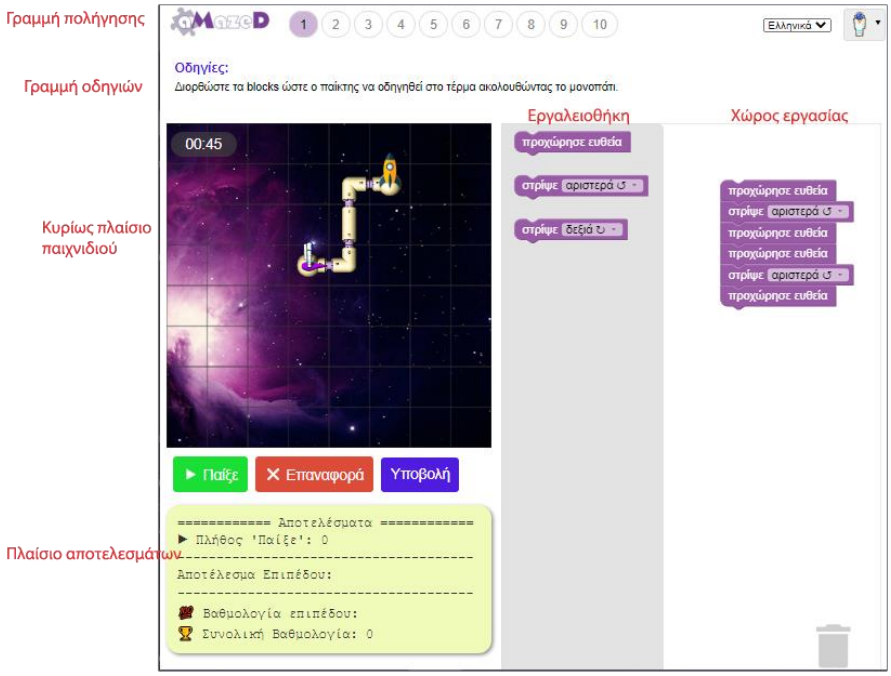
Τα επίπεδα του παιχνιδιού είναι δέκα και χωρίζονται σε δυο κατηγορίες: τον Λαβύρινθο και την Ζωγραφική. Στα πρώτα 6 επίπεδα (Λαβύρινθος) ο χρήστης προσπαθεί με την βοήθεια των μπλοκ να οδηγήσει τον χαρακτήρα του από την αφετηρία στο τέρμα, μέσω ενός λαβυρίνθου ενώ στα επόμενα 4 επίπεδα (Ζωγραφική) ο χρήστης καλείται να χρησιμοποιήσει τα μπλοκ για να δημιουργήσει ένα πρόγραμμα που θα σχεδιάσει το επιθυμητό σχήμα.

[8] Το λογότυπο σχεδιάστηκε και υλοποιήθηκε με το εργαλείο Gimp.

4.3. Περιγραφή διεπαφής χρήστη

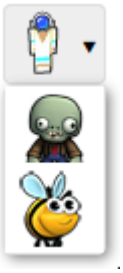
Το παιχνίδι aMazeD έχει μια πολύ απλή διεπαφή αποτελούμενη από μια γραμμή πλοήγησης, την γραμμή οδηγιών, το κυρίως πλαίσιο του παιχνιδιού, το πλαίσιο με τα αποτελέσματα του επιπέδου, την εργαλειοθήκη της Blockly και τον χώρο εργασίας.

Εικόνα 17: Διεπαφή παιχνιδιού



Η γραμμή πλοήγησης αποτελείται από το λογότυπο του παιχνιδιού το οποίο είναι υπερσύνδεση και ανακατευθύνει τον χρήστη στη σελίδα με τις πληροφορίες. Δεξιά του λογότυπου βρίσκονται οι αριθμοί των επιπέδων του παιχνιδιού. Με ανοιχτό μωβ χρώμα σημειώνονται τα ολοκληρωμένα επίπεδα καθώς και το τρέχον επίπεδο στο οποίο βρίσκεται ο χρήστης. Ο χρήστης σε αντίθεση με τα Παιχνίδια Blockly δεν έχει δικαίωμα μετάβασης σε επιθυμητό επίπεδο, ξεκινά από το επίπεδο 1 και με την υποβολή της απάντησης του μεταβαίνει σε κάθε επόμενο επίπεδο. Στο δεξιό κομμάτι της γραμμής πλοήγησης ο χρήστης μπορεί να επιλέξει την επιθυμητή γλώσσα (Ελληνικά - Αγγλικά) και στα επίπεδα του Λαβυρίνθου μπορεί να επιλέξει διαφορετική εμφάνιση επιπέδου ενώ στα επίπεδα της Ζωγραφικής μπορεί να δει την προτεινόμενη βοήθεια σε κάθε επίπεδο.

Εικόνα 18: Γραμμή πλοήγησης Λαβυρίνθου - Επιλογές εμφάνισης



Εικόνα 19: Γραμμή πλοήγησης επίπεδα 7-10

Η γραμμή οδηγιών έχει μια εισαγωγική εκφώνηση για το κάθε επίπεδο που ενημερώνει το χρήστη για το τι πρέπει να κάνει στο εκάστοτε επίπεδο.

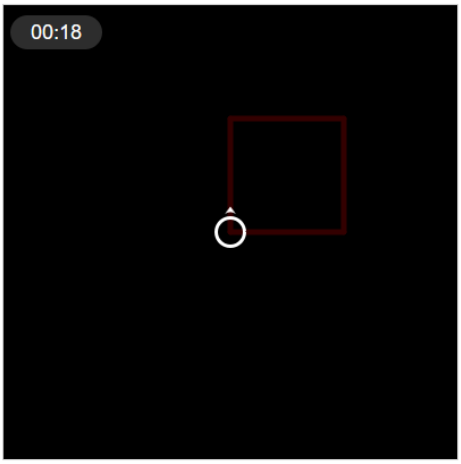
Το κυρίως πλαίσιο του παιχνιδιού είναι μια περιοχή 400 x 400 εικονοστοιχείων στην οποία ο χρήστης βλέπει το αποτέλεσμα της εκτέλεσης του κώδικα του. Το κυρίως πλαίσιο διαφοροποιείται σε εμφάνιση στα επίπεδα 1-6 και 7-10. Τα επίπεδα του Λαβυρίνθου (επίπεδα 1-6) αποτελούνται από τον χαρακτήρα του παίκτη, το αντίστοιχο φόντο με το μονοπάτι και το εικονίδιο τερματισμού, σε αυτά τα χαρακτηριστικά θα αναφερόμαστε με την φράση «εμφάνιση επιπέδου» (skin). Ο παίκτης έχει 3 επιλογές εμφάνισης επιπέδου

- Αστροναύτης
- Ζόμπι
- Μέλισσα

Εικόνα 20: Το επίπεδο 3 με διαφορετική εμφάνιση.
αριστερά : Αστροναύτης, κέντρο: Ζόμπι, δεξιά: Μέλισσα

Στα επίπεδα της Ζωγραφικής (επίπεδα 7-10) το φόντο του κυρίως πλαισίου είναι μαύρο και με αχνή γραμμή είναι προσχεδιασμένο το σχήμα το οποίο πρέπει να σχεδιάσει το πρόγραμμα του χρήστη. Δεν υπάρχει χαρακτήρας αλλά το πινέλο το οποίο βρίσκεται στην αρχή του σχήματος και είναι ένας λευκός κύκλος με ένα βέλος που δείχνει την κατεύθυνση.

Εικόνα 21 Κυρίως πλαίσιο επίπεδα Ζωγραφικής



Στην επάνω αριστερά γωνία του κυρίως πλαισίου βρίσκεται το χρονόμετρο κάθε επιπέδου που ξεκινά με την ανακατεύθυνση του χρήστη στο επίπεδο και σταματά με πάτημα του κουμπιού «Υποβολή».

Εικόνα 22 Πλήκτρα Παιχνιδιού

Κάτω από το κυρίως πλαίσιο βρίσκονται τα 3 πλήκτρα «Παίξε», «Επαναφορά», «Υποβολή».

Πλήκτρο «Παίξε»:

Ο παίκτης μπορεί να δει την κίνηση του χαρακτήρα ή του πινέλου ανάλογα με τον κώδικά του. Κατά τη διάρκεια εκτέλεσης του κώδικα επισημαίνονται τα μπλοκ του χώρου εργασίας των οποίων ο κώδικας εκτελείται. Μετά την ολοκλήρωση της εκτέλεσης δεν εμφανίζεται κάποιο αποτέλεσμα. Η λογική του πλήκτρου «Παίξε» είναι να δει ο παίκτης την κίνηση που έχει προγραμματίσει έχει δηλαδή το ρόλο της αποσφαλμάτωσης του κώδικα.

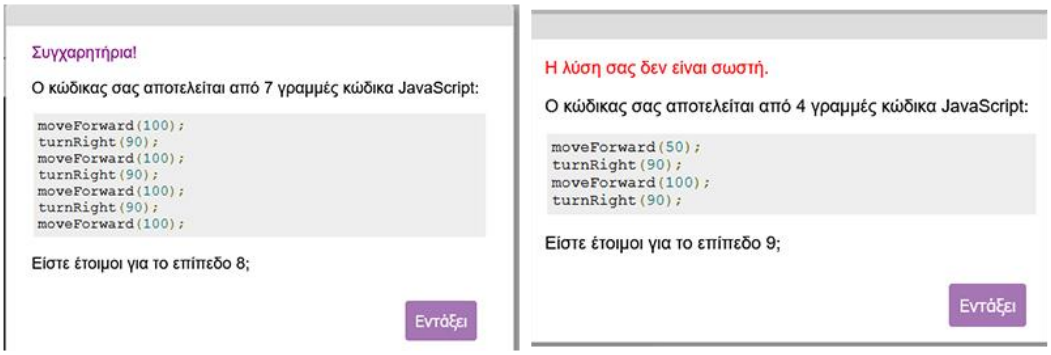
Πλήκτρο «Επαναφορά»:

Ο χαρακτήρας ή το πινέλο μετακινείται στην αρχή του μονοπατιού ή στην αρχή του σχήματος χωρίς να εκτελεστεί ο κώδικας του παίκτη. Γίνεται ουσιαστικά επαναφορά του παιχνιδιού στην αρχική κατάστασή του.

Πλήκτρο «Υποβολή»:

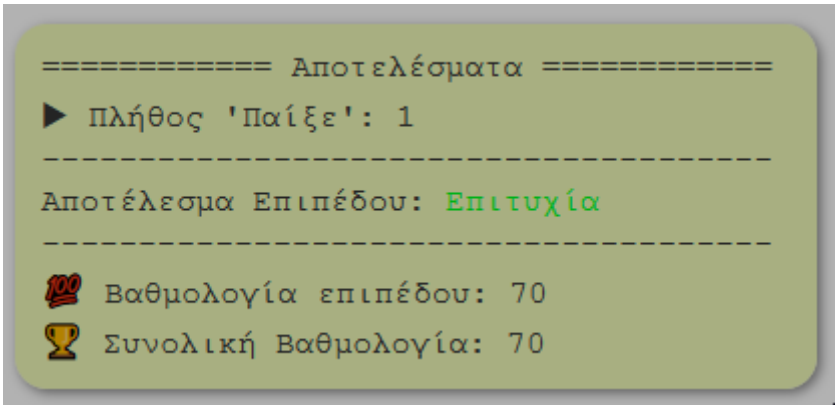
Ο παίκτης μπορεί να δει την κίνηση του χαρακτήρα ή του πινέλου ανάλογα με τον κώδικά του. Αν ο κώδικάς του είναι σωστός στα επίπεδα του Λαβυρίνθου η κίνηση είναι πιο γρήγορη, ενώ μειώνεται η ταχύτητα αν ο κώδικας είναι λανθασμένος. Κατά τη διάρκεια εκτέλεσης του κώδικα επισημαίνονται τα μπλοκ του χώρου εργασίας των οποίων ο κώδικας εκτελείται. Μετά την ολοκλήρωση της εκτέλεσης εμφανίζεται μήνυμα με το αποτέλεσμα του προγράμματος του παίκτη. Αν ο παίκτης κατάφερε να επιλύσει το ζητούμενο του επιπέδου εμφανίζεται μήνυμα επιτυχίας, σε αντίθετη περίπτωση εμφανίζεται μήνυμα αποτυχίας. Και στις δυο περιπτώσεις ο χρήστης μπορεί να δει τον κώδικα που έχει δημιουργήσει με τα μπλοκ και μεταφέρεται στο επόμενο επίπεδο. Η λογική του παιχνιδιού είναι ότι ο παίκτης πηγαίνει στο επόμενο επίπεδο ασχέτως με το αν έχει ολοκληρώσει επιτυχώς το τρέχον επίπεδο αυτή είναι και η λειτουργία του πλήκτρου «Υποβολή» να υποβάλλει δηλαδή ο παίκτης την απάντηση του όποια και να είναι.

Εικόνα 23: Αναδυόμενα μηνύματα ολοκλήρωσης επιπέδου



Το πλαίσιο αποτελεσμάτων παρουσιάζει κάποιες πληροφορίες για το τρέχον επίπεδο όπως είναι ο αριθμός των φορών που πατήθηκε το πλήκτρο «Παίξε», το αποτέλεσμα του επιπέδου (Επιτυχία ή Αποτυχία), η βαθμολογία του επιπέδου και τη συνολική βαθμολογία του παίκτη μέχρι και αυτό το επίπεδο. Το αποτέλεσμα και η βαθμολογία του επιπέδου εμφανίζονται μετά την υποβολή του αποτελέσματος από τον παίκτη. Ο παίκτης μπορεί να τα δει με μια διακριτική σκιά πίσω από το αναδυόμενο μήνυμα ολοκλήρωσης επιπέδου.

Εικόνα 24: Πλαίσιο αποτελεσμάτων

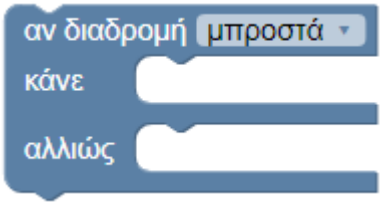
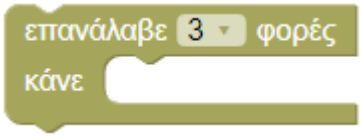
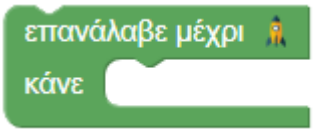


Η εργαλειοθήκη της Blockly περιέχει τα μπλοκ που είναι διαθέσιμα σε κάθε επίπεδο. Η εργαλειοθήκη διαφέρει στον Λαβύρινθο και στην Ζωγραφική. Στον Λαβύρινθο τα διαθέσιμα ανά επίπεδο μπλοκ είναι το ένα κάτω από το άλλο χωρίς κατηγορίες, ενώ στη Ζωγραφική τα μπλοκ είναι ίδια για όλα τα επίπεδα και χωρίζονται σε 3 κατηγορίες

- Κινήσεις: περιέχουν τα μπλοκ «Κινήσου μπροστά κατά …», «Στρίψε αριστερά κατά …μοίρες» , «στυλό πάνω»
- Χρώμα: περιέχει το μπλοκ «Κάνε το χρώμα …»
- Επαναλήψεις: περιέχει το μπλοκ «Επανέλαβε … φορές»

Πίνακας 7: Εργαλειοθήκη παιχνιδιού

Μπλοκ	**Όνομα και περιγραφή**
προχώρησε ευθεία	**Προχώρησε ευθεία** Ο χαρακτήρας προχωρά ευθεία για ένα βήμα.
στρίψε αριστερά ↺	**Στρίψε αριστερά / δεξιά** Ο χαρακτήρας στρίβει επί τόπου αριστερά ή δεξιά ανάλογα με την επιλογή του χρήστη.
αν διαδρομή μπροστά κάνε	**Αν υπάρχει διαδρομή μπροστά/αριστερά/δεξιά** Αν υπάρχει διαδρομή στην κατεύθυνση που επιλέγει ο χρήστης εκτελείται το μπλοκ που βρίσκεται μέσα στο "κάνε". Για παράδειγμα αν μπει το μπλοκ "προχώρησε μπροστά" ο

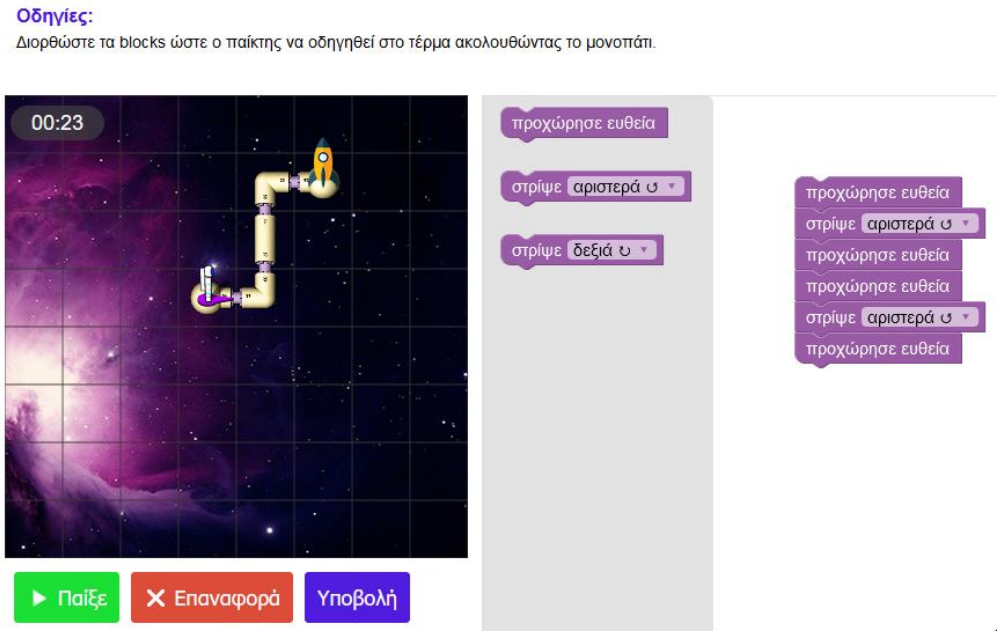
	παίκτης θα προχωρήσει ένα βήμα μπροστά όταν υπάρχει διαδρομή (μπροστά/αριστερά/δεξιά) ανάλογα με την επιλογή.
	**Αν υπάρχει διαδρομή μπροστά/αριστερά/δεξιά ..(συνθήκη)..αλλιώς..(εναλλακτική συνθήκη)** Αν υπάρχει διαδρομή προς την κατεύθυνση που επιλέγει ο χρήστης εκτελείται το μπλοκ στο "κάνε". Αν δεν υπάρχει διαδρομή προς αυτή την κατεύθυνση εκτελείται το μπλοκ στο "αλλιώς".
	**Επανέλαβε ν-φορές** Επαναλαμβάνεται το μπλοκ μέσα στο "κάνε" όσες φορές επιλέγει ο χρήστης.
	**Επανέλαβε μέχρι...τέρμα** Επαναλαμβάνεται το μπλοκ στο "κάνε" μέχρι να φτάσει ο παίκτης στο τέρμα.
	**Κινήσου μπροστά κατά κ - εικονοστοιχεία** Το πινέλο κινείται μπροστά κατά το ορισμένο ποσό (οι μονάδες μέτρησης είναι σε pixel).
	**Στρίψε δεξιά/αριστερά κατά μ-μοίρες** Το πινέλο στρίβει δεξιά/αριστερά ανάλογα με την επιλογή του χρήστη, τόσες μοίρες όσες επιλέγει ο χρήστης από το δεύτερο πτυσσόμενο μενού επιλογής.
	**Κάνε το χρώμα...** Αλλάζει το χρώμα του πινέλου ανάλογα με την επιλογή του χρήστη.

Στον χώρο εργασίας ο παίκτης μπορεί να στοιβάξει τα μπλοκ του ώστε να δημιουργήσει το πρόγραμμα που θα επιλύσει το επίπεδο. Οι δυο εκδόσεις του παιχνιδιού aMazeD έχουν μια βασική διαφορά, στην έκδοση 1 (ή πρώτη έκδοση) το παιχνίδι έχει προ συμπληρωμένο κώδικα στο χώρο εργασίας δηλαδή υπάρχουν κάποια μπλοκ που συνθέτουν τον κώδικα και ο παίκτης καλείται είτε να τα διορθώσει είτε να τα συμπληρώσει είτε να τα αναδιατάξει. Επιπλέον στην πρώτη έκδοση γίνεται ανατροφοδότηση του χρήστη με συμβουλές κατά την διάρκεια του επιπέδου. Στην δεύτερη έκδοση του παιχνιδιού ο χώρος εργασίας είναι άδειος και στην εργαλειοθήκη ο χρήστης βλέπει τα μπλοκ που μπορεί να χρησιμοποιήσει για να δημιουργήσει το πρόγραμμά του από το μηδέν. Δεν δίνεται κάποια βοήθεια στον χρήστη πέρα από μια τυπική εκφώνηση στη γραμμή οδηγιών.

4.4. Περιγραφή επιπέδων

4.4.1. Επίπεδο 1 – Λαβύρινθος

Εικόνα 25: Επίπεδο 1 - έκδοση 1



Στην πρώτη έκδοση του επιπέδου εμφανίζεται η εκφώνηση «Διορθώστε τα blocks ώστε ο παίκτης να οδηγηθεί στο τέρμα ακολουθώντας το μονοπάτι.». Στον χώρο εργασίας βρίσκεται προ συμπληρωμένο ένα κομμάτι του κώδικα όπως φαίνεται στην Εικόνα 25. Ο χρήστης πρέπει να βρει το λάθος στον υπάρχων κώδικα.

Η πρώτη συμβουλή είναι «Ένα από τα βήματα είναι λάθος. Αφού δοκιμάσετε να εκτελέσετε τον κώδικα αλλάξτε το βήμα που πρέπει για να φτάσω στο τέρμα.» και εμφανίζεται με την έναρξη του επιπέδου.

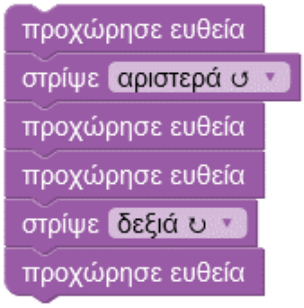
Η δεύτερη συμβουλή είναι «Κάνοντας κλικ στο βέλος του block 'στρίψε' μπορείτε να αλλάξετε την κατεύθυνση που στρίβει ο παίκτης.» και εμφανίζεται αφού ο παίκτης έχει τρέξει τον κώδικα μια φορά με το πλήκτρο «Παίξε» αλλά το πρόγραμμά του δεν επιλύει το επίπεδο.

Η τρίτη συμβουλή είναι «Κάποια από τις στροφές που παίρνει ο παίκτης είναι λάθος.» και εμφανίζεται όταν ο παίκτης πατήσει το πλήκτρο «Παίξε» περισσότερες από δυο φορές και ο κώδικάς του δεν είναι σωστός.

Εικόνα 26: Επίπεδο 1 - Έκδοση 2

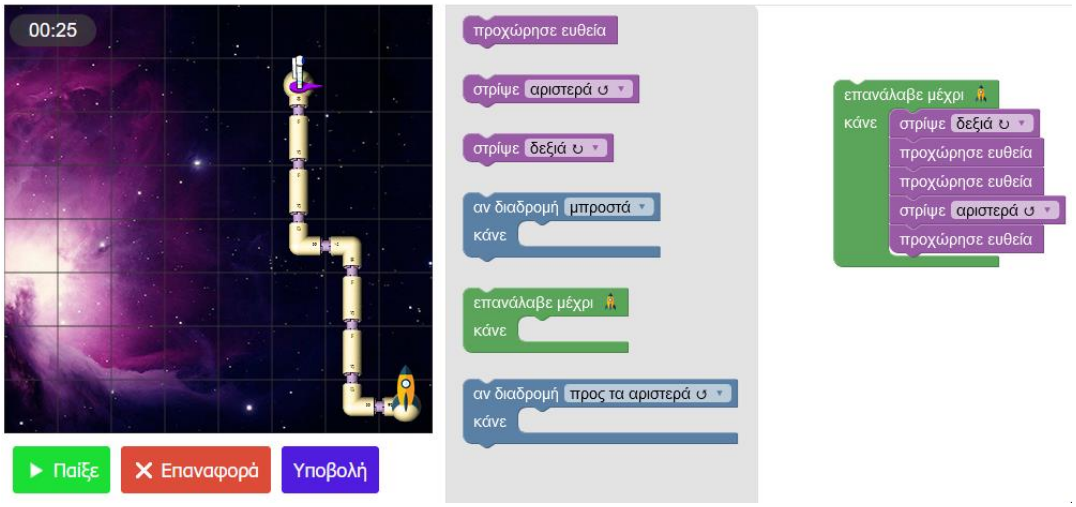
Στην δεύτερη έκδοση εμφανίζονται στην εργαλειοθήκη μόνο τα μπλοκ που φαίνονται στην εικόνα 19. Δίνεται η εκφώνηση «Δώστε τις εντολές ώστε ο παίχτης να οδηγηθεί στο τέρμα ακολουθώντας το μονοπάτι.» και δεν δίνεται καμία άλλη συμβουλή.

Λύση επιπέδου για έκδοση 1 και προτεινόμενη λύση για έκδοση 2:



4.4.2. Επίπεδο 2 – Λαβύρινθος

Εικόνα 27: Επίπεδο 2 - έκδοση 1



Στο επίπεδο 2 στην πρώτη έκδοση δίνεται η εκφώνηση «Διορθώστε τα blocks ώστε ο παίκτης να οδηγηθεί στο τέρμα ακολουθώντας το μονοπάτι.» και στο χώρο εργασίας εμφανίζεται ένα κομμάτι του προγράμματος όπως φαίνεται στην Εικόνα 27.

Δύο δευτερόλεπτα μετά την έναρξη του επιπέδου εμφανίζεται η συμβουλή «Το 'επανέλαβε' χρησιμοποιείται για να εκτελεστούν ένα ή περισσότερα blocks περισσότερες από μια φορές.».

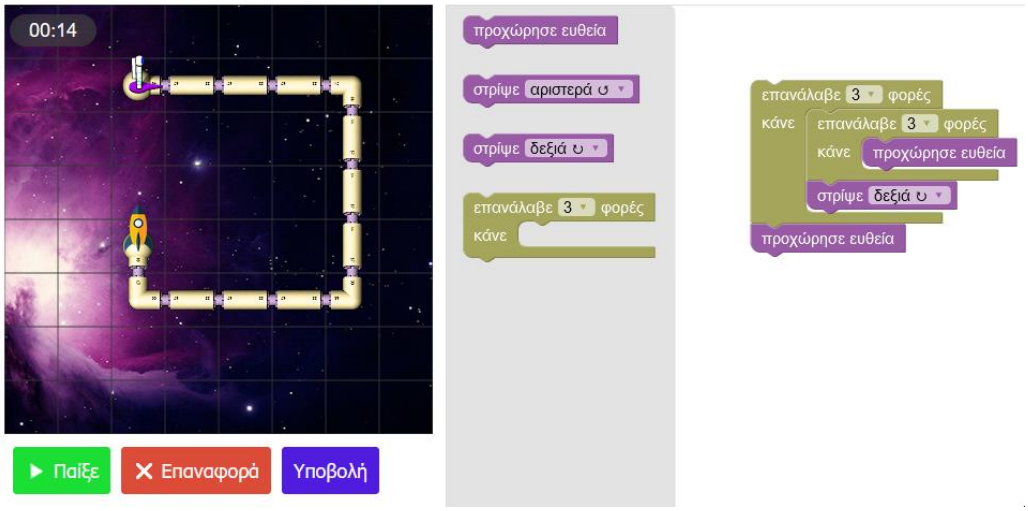
Η δεύτερη συμβουλή είναι «Πόσα βήματα πρέπει να κάνει ο παίκτης πριν στρίψει αριστερά; Πόσα κάνει τώρα;» και εμφανίζεται αφού ο παίκτης έχει εκτελέσει τον κώδικά του ανεπιτυχώς μια φορά με το πλήκτρο «Παίξε».

Στην δεύτερη έκδοση εμφανίζεται η εκφώνηση «Δώστε τις εντολές ώστε ο παίχτης να οδηγηθεί στο τέρμα ακολουθώντας το μονοπάτι.» χωρίς καμία επιπλέον βοήθεια και χωρίς να υπάρχει κάτι στον χώρο εργασίας. Στην εργαλειοθήκη εμφανίζονται τα μπλοκ «προχώρησε ευθεία» , «στρίψε αριστερά/ δεξιά» , «αν διαδρομή μπροστά / προς τα αριστερά/ προς τα δεξιά», «επανέλαβε μέχρι τέλος» .

Λύση επιπέδου

4.4.3. Επίπεδο 3 – Λαβύρινθος

Εικόνα 28: Επίπεδο 3 - έκδοση 1



Στο τρίτο επίπεδο στην έκδοση με την βοήθεια εμφανίζεται η εκφώνηση «Διόρθωσε τις επιλογές των blocks ώστε ο παίκτης να οδηγηθεί στο τέρμα ακολουθώντας το μονοπάτι.» και στον χώρο εργασίας βρίσκεται το κομμάτι του κώδικα όπως φαίνεται στην Εικόνα 28.
Η πρώτη συμβουλή εμφανίζεται όταν ο παίκτης εκτελέσει την λύση του με το πλήκτρο «Παίξε» και δεν είναι σωστή, η συμβουλή είναι «Ένα επανέλαβε μπορεί να μπει μέσα σε ένα άλλο.» .
Η δεύτερη συμβουλή είναι «Πόσα βήματα πρέπει να κάνει ο παίκτης για να καλύψει κάθε πλευρά του μονοπατιού και άρα πόσες φορές πρέπει να εκτελεστεί το block 'προχώρησε ευθεία'; Σε ποιο από τα δυο επανέλαβε πρέπει να αλλάξετε τον αριθμό των φορών που εκτελούνται τα blocks;» και εμφανίζεται όταν ο παίκτης πατήσει το πλήκτρο «Παίξε» 2 ή περισσότερες φορές χωρίς να είναι σωστή η λύση του.

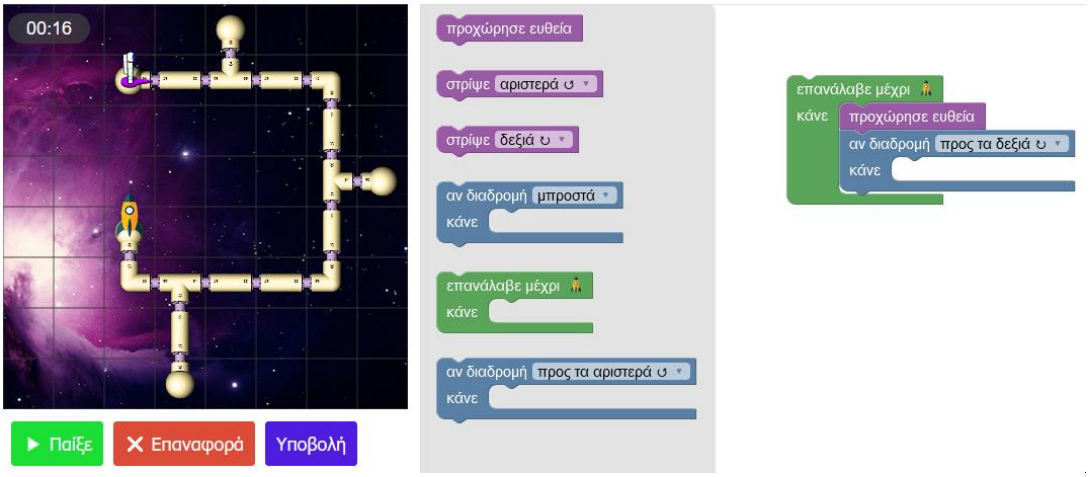
Στην έκδοση 2 χωρίς βοήθεια εμφανίζεται η εκφώνηση «Δώστε τις εντολές ώστε ο παίχτης να οδηγηθεί στο τέρμα ακολουθώντας το μονοπάτι.» και δεν δίνεται περεταίρω βοήθεια ενώ στην εργαλειοθήκη βρίσκονται τα μπλοκ «προχώρησε ευθεία», «στρίψε αριστερά / δεξιά», «επανέλαβε 3/4/5 φορές».

Η λύση του επιπέδου είναι

4.4.4. Επίπεδο 4 – Λαβύρινθος

Στην πρώτη έκδοση στο επίπεδο 4 εμφανίζεται η εκφώνηση «Συμπληρώστε τα blocks έτσι ώστε ο παίκτης να οδηγηθεί στο τέρμα ακολουθώντας το μονοπάτι.». Στο χώρο εργασίας βρίσκονται τα μπλοκ όπως φαίνονται στην Εικόνα 29.

Εικόνα 29: Επίπεδο 4 - έκδοση 1

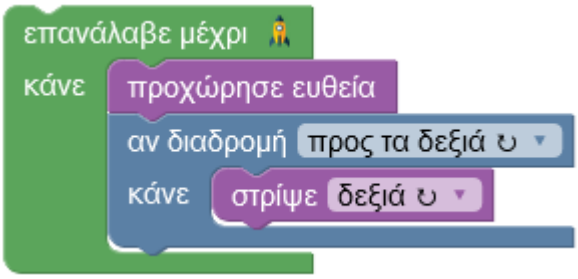


Η πρώτη συμβουλή είναι «Ένα μπλοκ «Αν» θα κάνει κάτι μόνο αν η συνθήκη είναι αληθής.» και εμφανίζεται όταν ο παίκτης πατήσει το πλήκτρο «Παίξε» και η λύση του δεν είναι σωστή.

Η δεύτερη συμβουλή είναι «Δοκιμάστε να στρίψει ο παίκτης δεξιά αν υπάρχει διαδρομή προς τα δεξιά.» και εμφανίζεται όταν ο παίκτης δοκιμάσει τον κώδικά του με το πλήκτρο παίξε 2 ή περισσότερες φορές χωρίς να έχει σωστή λύση.

Στην έκδοση 2 χωρίς βοήθεια εμφανίζεται η εκφώνηση «Δώστε τις εντολές ώστε ο παίχτης να οδηγηθεί στο τέρμα ακολουθώντας το μονοπάτι.» και δεν δίνεται περεταίρω βοήθεια ενώ στην εργαλειοθήκη βρίσκονται τα μπλοκ «προχώρησε ευθεία», «στρίψε αριστερά / δεξιά», «επανέλαβε 3/4/5 φορές».

Η λύση του επιπέδου είναι



4.4.5. Επίπεδο 5 – Λαβύρινθος

Στο επίπεδο 5 στην πρώτη έκδοση δίνεται η εκφώνηση «Διορθώστε τα blocks ώστε ο παίκτης να οδηγηθεί στο τέρμα ακολουθώντας το μονοπάτι.» ενώ στο χώρο εργασίας εμφανίζονται τα μπλοκ όπως φαίνονται στην Εικόνα 30.

Εικόνα 30: Επίπεδο 5 - πρώτη έκδοση

Πρώτη συμβουλή «Το 'επανάλαβε μέχρι' χρησιμοποιείται για να εκτελεστούν ένα ή περισσότερα blocks μέχρι μια συνθήκη να γίνει αληθής, για παράδειγμα μέχρι να φτάσει ο παίκτης στο τέρμα.» , εμφανίζεται όταν ο παίκτης πατήσει το πλήκτρο «Παίξε» μια φορά χωρίς να έχει σωστή λύση.

Δεύτερη συμβουλή «Δοκιμάστε να στρίψει ο παίκτης αριστερά εάν η διαδρομή είναι προς τα αριστερά. Αντίστοιχα προς τα που πρέπει να στρίψει ο παίκτης εάν η διαδρομή είναι

προς τα δεξιά;» εμφανίζεται όταν ο παίκτης πατήσει το πλήκτρο «Παίξε» περισσότερες από 2 φορές χωρίς να έχει σωστή λύση.

Η δεύτερη συμβουλή είναι «Δοκιμάστε να στρίψει ο παίκτης δεξιά αν υπάρχει διαδρομή προς τα δεξιά.» και εμφανίζεται όταν ο παίκτης δοκιμάσει τον κώδικά του με το πλήκτρο παίξε 2 ή περισσότερες φορές χωρίς να έχει σωστή λύση.

Στην έκδοση 2 χωρίς βοήθεια εμφανίζεται η εκφώνηση «Δώστε τις εντολές ώστε ο παίχτης να οδηγηθεί στο τέρμα ακολουθώντας το μονοπάτι.» και δεν δίνεται περεταίρω βοήθεια ενώ στην εργαλειοθήκη βρίσκονται τα μπλοκ «προχώρησε ευθεία», «στρίψε αριστερά / δεξιά», «επανέλαβε 3/4/5 φορές».

Η λύση του επιπέδου είναι

4.4.6. Επίπεδο 6 – Λαβύρινθος

Στο επίπεδο 6 στην πρώτη έκδοση εμφανίζεται η εκφώνηση «Συμπληρώστε την εντολή στο αλλιώς ώστε ο παίκτης να οδηγηθεί στο τέρμα ακολουθώντας το μονοπάτι.» και τα μπλοκ στον χώρο εργασίας όπως φαίνεται στην Εικόνα 31.

Εικόνα 31: Επίπεδο 6

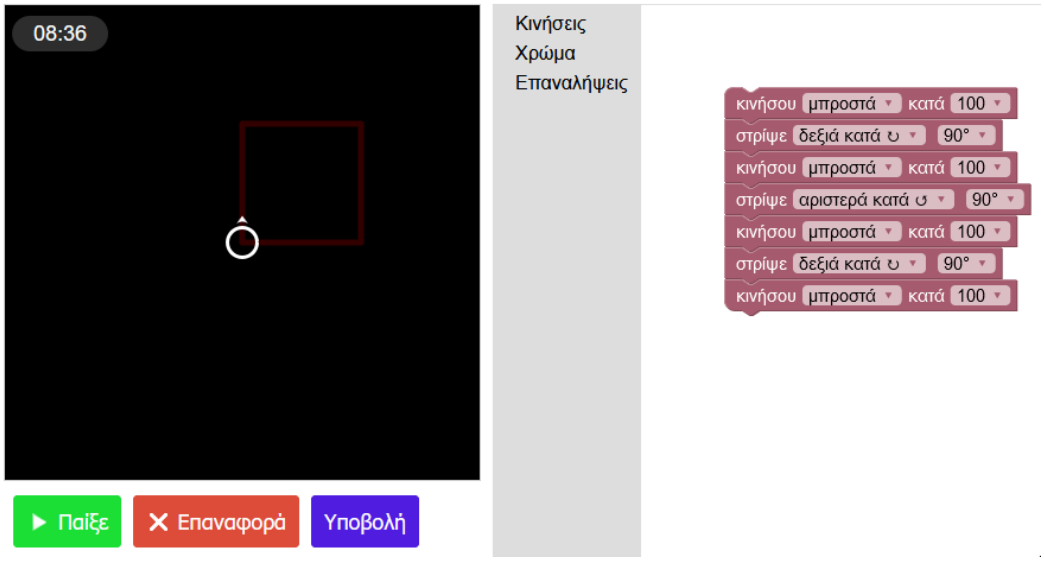
Πρώτη συμβουλή «Στο block 'Αν-αλλιώς' εκτελείται το 'κάνε' ή το 'αλλιώς' . Το 'κάνε' εκτελείται αν η συνθήκη είναι αληθής, ενώ το 'αλλιώς' αν η συνθήκη είναι ψευδής. Επιπλέον, ένα block 'Αν-αλλιώς' μπορεί να μπει μέσα σε ένα άλλο.», εμφανίζεται όταν ο παίκτης πατήσει το πλήκτρο «Παίξε» μια φορά.

Δεύτερη συμβουλή «Δοκιμάστε να στρίψει ο παίκτης δεξιά εάν υπάρχει διαδρομή προς τα δεξιά. Αλλιώς αν η διαδρομή είναι προς τα αριστερά ποιο block πρέπει να εκτελεστεί;» εμφανίζεται όταν ο παίκτης δοκιμάσει να εκτελέσει τον κώδικά του 2 ή περισσότερες φορές με το πλήκτρο «Παίξε» .

Στην έκδοση 2 χωρίς βοήθεια εμφανίζεται η εκφώνηση «Δώστε τις εντολές ώστε ο παίχτης να οδηγηθεί στο τέρμα ακολουθώντας το μονοπάτι.» και δεν δίνεται περεταίρω βοήθεια ενώ στην εργαλειοθήκη βρίσκονται τα μπλοκ «προχώρησε ευθεία», «στρίψε αριστερά/δεξιά», «επανέλαβε 3/4/5 φορές» και «αν διαδρομή…κάνε…αλλιώς». Η λύση είναι

4.4.7. Επίπεδο 7 – Ζωγραφική

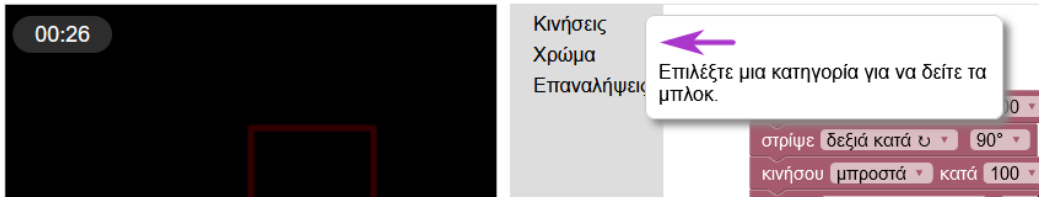
Εικόνα 32: Επίπεδο 7 - πρώτη έκδοση



Στο επίπεδο 7 στην πρώτη έκδοση εμφανίζεται η εκφώνηση «Διορθώστε τα blocks έτσι ώστε να σχεδιαστεί ένα τετράγωνο με πλευρά 100.». Στον χώρο εργασίας εμφανίζεται προ συμπληρωμένο το πρόγραμμα όπως φαίνεται στην Εικόνα 32.

Επειδή είναι το πρώτο επίπεδο στο οποίο εμφανίζονται οι κατηγορίες των μπλοκ στον χρήστη εμφανίζεται το αναδυόμενο παράθυρο της Εικόνας 33 που τον ενημερώνει για τις κατηγορίες. Το μήνυμα κλείνει μόνο όταν ο χρήστης επιλέξει κάποια κατηγορία.

Εικόνα 33: Αναδυόμενο παράθυρο βοήθειας επιπέδου 7

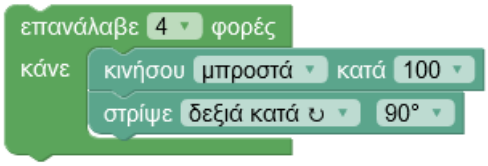


Μετά την πρώτη εκτέλεση του κώδικα με το πλήκτρο «Παίξε», εμφανίζεται η συμβουλή «Δοκιμάστε να στρίβετε πάντα προς την ίδια κατεύθυνση.» .

Στην έκδοση χωρίς συμβουλές δεν εμφανίζεται κάποιο μπλοκ στον χώρο εργασίας και ο χρήστης δεν δέχεται κάποια επιπλέον βοήθεια πέρα από την εκφώνηση «Δώστε τα blocks έτσι ώστε να σχεδιαστεί ένα τετράγωνο με πλευρά 100.».

Η λύση του επιπέδου είναι

η βέλτιστη λύση για την έκδοση χωρίς βοήθεια είναι

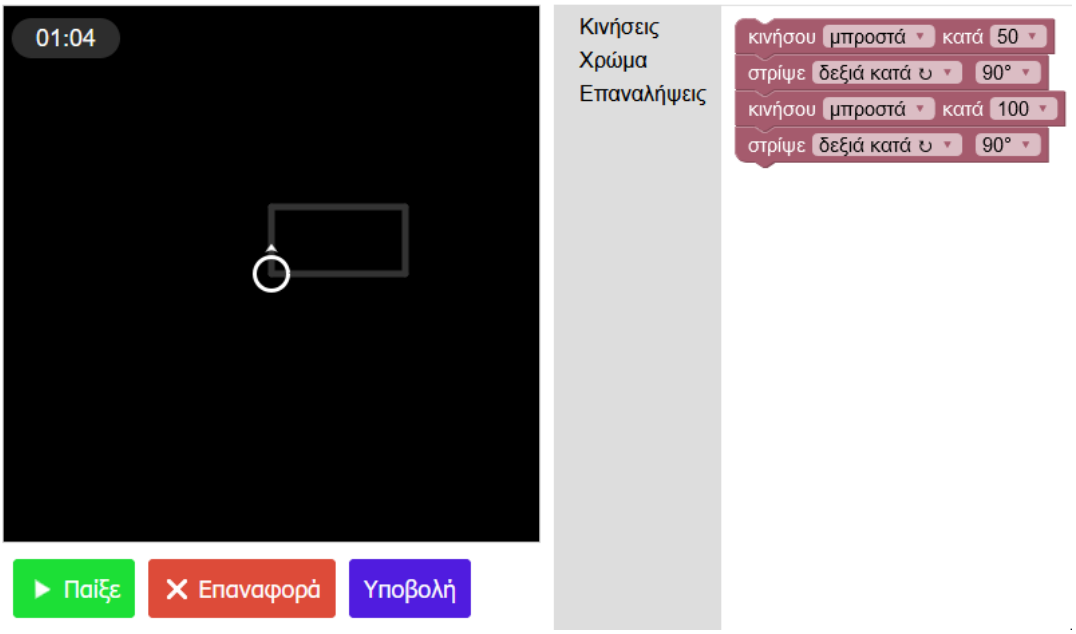


4.4.8. Επίπεδο 8 - Ζωγραφική

Σε αυτό το επίπεδο ο χρήστης καλείται να σχηματίσει ένα ορθογώνιο διαστάσεων 50x100. Στην πρώτη έκδοση δίνεται η εκφώνηση «Συμπληρώστε το πρόγραμμα έτσι ώστε να

σχεδιαστεί ένα ορθογώνιο με πλάτος 100 και ύψος 50.» και στον χώρο εργασίας βρίσκονται τα μπλοκ όπως φαίνεται στην Εικόνα 34.

Εικόνα 34: Επίπεδο 8 - πρώτη έκδοση



Μετά την εκτέλεση του κώδικα με το «Παίξε» και δεδομένου ότι δεν επιλύει το επίπεδο, εμφανίζεται η συμβουλή «Παρατηρήστε ποιο τμήμα του σχήματος δημιουργείται αν οι εντολές που δίνονται εκτελεστούν μία φορά. Πόσες φορές πρέπει να εκτελεστούν για να δημιουργηθεί το ζητούμενο ορθογώνιο;». Στόχος αυτού του επιπέδου είναι ο παίκτης να χρησιμοποιήσει το μπλοκ «Επανέλαβε … φορές» με το οποίο έχει εξοικειωθεί σε προηγούμενα επίπεδα.

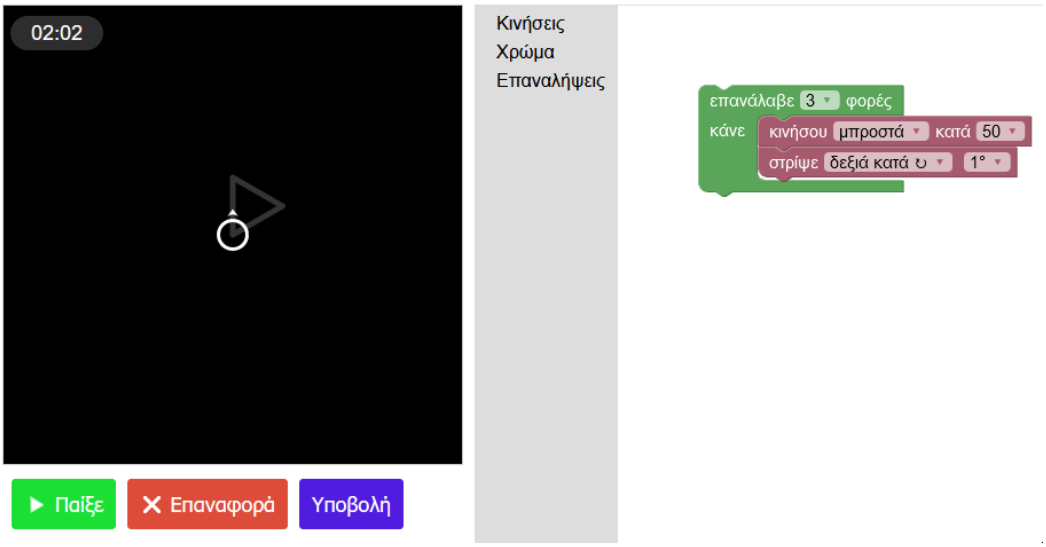
Στην έκδοση 2 εμφανίζεται απλά η εκφώνηση «Δώστε τα blocks έτσι ώστε να σχεδιαστεί ένα ορθογώνιο με ύψος 50 και πλάτος 100.».

Η λύση του επιπέδου είναι

4.4.9. Επίπεδο 9 – Ζωγραφική

Στο επίπεδο 9 ο χρήστης πρέπει να δημιουργήσει ένα ισόπλευρο τρίγωνο πλευράς 50. Στην πρώτη έκδοση δίνεται η εκφώνηση «Συμπληρώστε το πρόγραμμα ώστε να σχεδιαστεί ένα ισόπλευρο τρίγωνο με πλευρά 50.» και στον χώρο εργασίας υπάρχουν τα μπλοκ όπως φαίνονται στην Εικόνα 35.

Εικόνα 35: Επίπεδο 9 - πρώτη έκδοση



Η συμβουλή του επιπέδου είναι «Οι γωνίες ενός ισόπλευρου τριγώνου είναι 60 μοίρες. Για να σχηματιστεί όμως το σχήμα πρέπει να στρίψουμε όσο είναι η παραπληρωματική γωνία. Δηλαδή 180-60=?» και εμφανίζεται αφού ο παίκτης πατήσει το πλήκτρο «Παίξε» . Στόχος είναι ο παίκτης να διορθώσει τις μοίρες στο στρίψε και να τις κάνει 120.

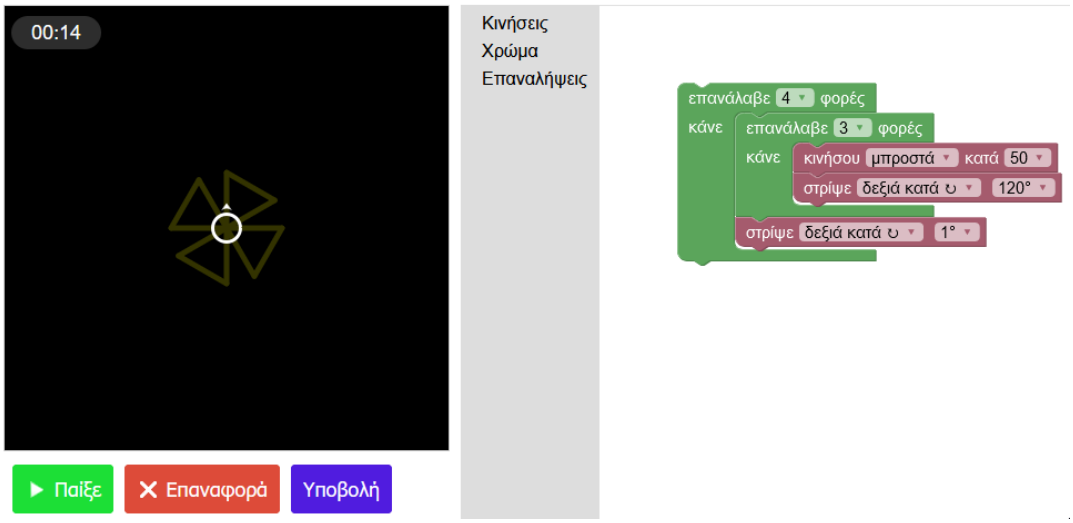
Στην έκδοση 2 δίνεται μόνο η εκφώνηση «Δώστε τα blocks έτσι ώστε να σχεδιαστεί ένα ορθογώνιο με ύψος 50 και πλάτος 100.» .

Λύση επιπέδου

4.4.10. Επίπεδο 10 – Ζωγραφική

Στο επίπεδο 10 ο παίκτης καλείται να σχηματίσει το σχήμα του κυρίως πλαισίου.
Στην πρώτη έκδοση δίνεται η εκφώνηση «Διορθώστε τα blocks έτσι ώστε να σχεδιαστεί το σχήμα που εμφανίζεται.» και στον χώρο εργασίας εμφανίζονται τα μπλοκ όπως φαίνονται στην Εικόνα 36.

Εικόνα 36: Επίπεδο 10 - πρώτη έκδοση



Μετά το πρώτο «Παίξε» εμφανίζεται στον χρήστη η συμβουλή «Το σχήμα αποτελείται από 4 ισόπλευρα τρίγωνα. Σκεφτείτε πόσες μοίρες πρέπει να στρίβει το πινέλο μετά τη σχεδίαση κάθε τριγώνου.». Στόχος του επιπέδου είναι να αλλάξει ο παίκτης τις μοίρες του τελευταίου στρίψε σε 90.
Στην έκδοση 2 δεν δίνεται κάποια βοήθεια στον χρήστη πέρα από την εκφώνηση «Δώστε τα blocks έτσι ώστε να σχεδιαστεί το σχήμα που εμφανίζεται.».
Η λύση του επιπέδου είναι

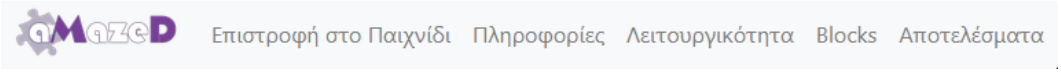
4.5. Βοηθητικές σελίδες παιχνιδιού

Το παιχνίδι aMazeD διαθέτει δυο βοηθητικές σελίδες την σελίδα πληροφοριών (about) και την σελίδα αποτελεσμάτων (result). Και οι δυο σελίδες είναι διαθέσιμες στα ελληνικά και στα αγγλικά και η ανακατεύθυνση γίνεται με βάση την τρέχουσα γλώσσα που έχει επιλέξει ο χρήστης. Οι δυο σελίδες έχουν δημιουργηθεί με την χρήση της βιβλιοθήκης Bootstrap 4.3.1 .

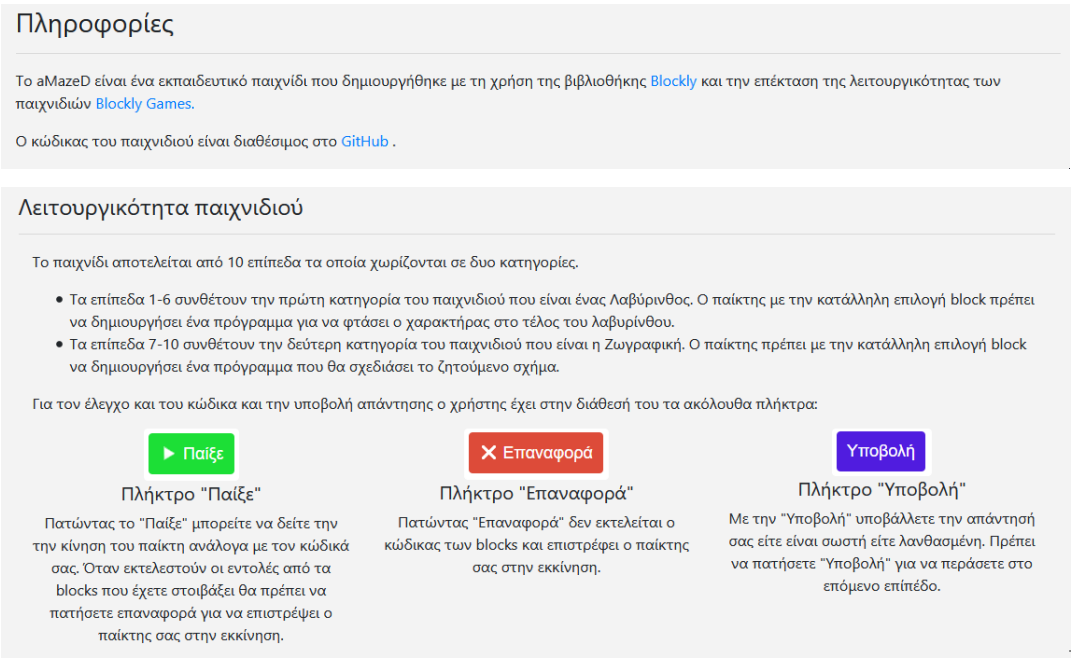
4.5.1. Σελίδα Πληροφορίες (about.html)

Η σελίδα πληροφοριών ενημερώνει τον χρήστη για το παιχνίδι, τις λειτουργικότητες των πλήκτρων και των μπλοκ και παρέχει τις συνδέσεις για την βιβλιοθήκη της Blockly και των Παιχνιδιών Blockly στα οποία έχει στηριχτεί η ανάπτυξη του παιχνιδιού καθώς και ένα σύνδεσμο για τον κώδικα στο GitHub.

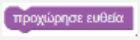
Εικόνα 37: Γραμμή πλοήγησης σελίδας Πληροφορίες



Εικόνα 38: Ενότητες της σελίδας Πληροφορίες

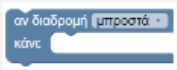


Blocks

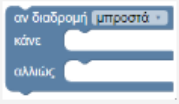


Προχώρησε ευθεία
Ο παίκτης προχωρά ευθεία για ένα βήμα.

Στρίψε αριστερά/δεξιά
Ο παίκτης στρίβει επί τόπου αριστερά ή δεξιά ανάλογα με την επιλογή του χρήστη.



Αν υπάρχει διαδρομή μπροστά/αριστερά/δεξιά
Αν υπάρχει διαδρομή στην κατεύθυνση που επιλέγει ο χρήστης εκτελείται το block που βρίσκεται μέσα στο "κάνε". Για παράδειγμα αν μπει το block "προχώρησε μπροστά" ο παίκτης θα προχωρήσει ένα βήμα μπροστά όταν υπάρχει διαδρομή (μπροστά/αριστερά/δεξιά) ανάλογα με την επιλογή.

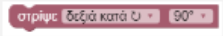


Αν υπάρχει διαδρομή μπροστά/αριστερά/δεξιά ..(συνθήκη)..αλλιώς..(εναλλακτική συνθήκη)
Αν υπάρχει διαδρομή προς την κατεύθυνση που επιλέγει ο χρήστης εκτελείται το block στο "κάνε". Αν δεν υπάρχει διαδρομή προς αυτή την κατεύθυνση εκτελείται το block στο "αλλιώς".

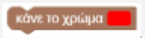
Επανέλαβε ν-φορές
Επαναλαμβάνεται το block μέσα στο "κάνε" όσες φορές επιλέγει ο χρήστης.

Επανέλαβε μέχρι...τέρμα
Επαναλαμβάνεται το block στο "κάνε" μέχρι να φτάσει ο παίκτης στο τέρμα.

Κινήσου μπροστά κ-pixel
Το πινέλο κινείται μπροστά κατά το ορισμένο ποσό (οι μονάδες μέτρησης είναι σε pixel).



Στρίψε δεξιά/αριστερά κατά μ-μοίρες
Το πινέλο στρίβει δεξιά/αριστερά ανάλογα με την επιλογή του χρήστη τόσες μοίρες όσες επιλέγει ο χρήστης από το δεύτερο πτυσσόμενο μενού επιλογής.



Κάνε το χρώμα...
Αλλάζει το χρώμα του πινέλου ανάλογα με την επιλογή του χρήστη.

Αποτελέσματα

Με την ολοκλήρωση του παιχνιδιού εμφανίζεται η σελίδα των αποτελεσμάτων με τα δεδομένα ανά επίπεδο και την συνολική βαθμολογία του παιχνιδιού. Για να εμφανιστούν τα αποτελέσματα είναι απαραίτητο να εισάγετε ένα όνομα και μια έγκυρη διεύθυνση e-mail.

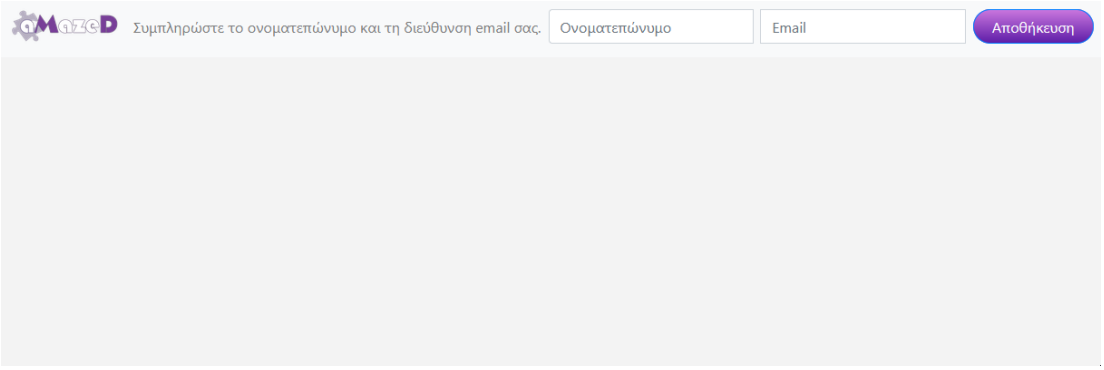
Γίνεται μόνο στατιστική χρήση των δεδομένων σας και με την επιλογή "Νέο Παιχνίδι" διαγράφονται αυτόματα. ×

Built on Blockly Blockly Games

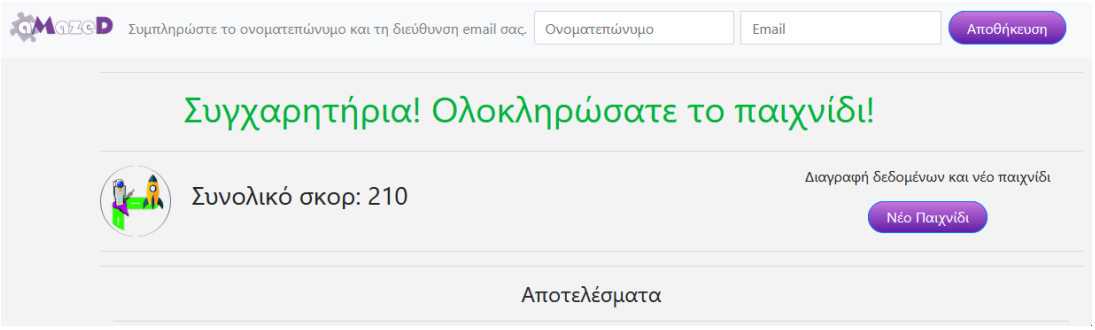
4.5.2. Σελίδα Αποτελέσματα (results.html)

Με την ολοκλήρωση του επιπέδου 10 το παιχνίδι ανακατευθύνει τον χρήστη στη σελίδα αποτελεσμάτων. Ο χρήστης πρέπει να δώσει ένα όνομα και ένα έγκυρο e-mail για να μπορέσει να δει τα αποτελέσματα, ο λόγος είναι ότι τα αποτελέσματα αποθηκεύονται τοπικά και το όνομα και το e-mail του χρήστη θα αποθηκευτούν μαζί με τα αποτελέσματα του ώστε να μπορεί να υπάρχει αντιστοίχιση.

Εικόνα 39: Σελίδα Αποτελέσματα - πριν την εισαγωγή δεδομένων χρήστη



Εικόνα 40: Σελίδα Αποτελέσματα - μετά την εισαγωγή δεδομένων χρήστη



Αφού ο χρήστης εισάγει τα βασικά δεδομένα που του ζητούνται ενημερώνεται με μήνυμα για την επιτυχή ολοκλήρωση του παιχνιδιού και για το συνολικό σκορ που έχει επιτύχει. Η αποθήκευση των δεδομένων του χρήστη ακολουθεί την λογική των παιχνιδιών Blockly και γίνεται τοπικά σε επίπεδο φυλλομετρητή, δηλαδή κάθε φορά που ο χρήστης ανοίγει τον ίδιο φυλλομετρητή θα επιστρέφει στην κατάσταση που είχε αφήσει το παιχνίδι, αν το παιχνίδι είχε ολοκληρωθεί στην τελευταία του επίσκεψη αλλά δεν έγινε διαγραφή δεδομένων θα δει τη σελίδα των αποτελεσμάτων. Πατώντας το πλήκτρο Νέο Παιχνίδι γίνεται διαγραφή των δεδομένων του φυλλομετρητή και ο παίκτης ξεκινά από το επίπεδο 1 του aMazeD.

Στην ενότητα αποτελέσματα ο χρήστης βλέπει έναν πίνακα με στήλες τα δεδομένα που διατηρούνται σε κάθε επίπεδο

- Επίπεδο
- Αποτέλεσμα : το αποτέλεσμα του επιπέδου Επιτυχία / Αποτυχία
- Βαθμολογία : η βαθμολογία του επιπέδου , μηδέν αν ήταν Αποτυχία
- Χρόνος : ο χρόνος που χρειάστηκε ο παίκτης μέχρι να πατήσει «Υποβολή»
- Πλήθος «Παίξε» : το πλήθος των πλήκτρων «Παίξε» που πάτησε ο παίκτης στο επίπεδο
- Κώδικας JavaScript : ο κώδικας που δημιούργησε ο παίκτης με τα μπλοκ στον χώρο εργασίας.

ο πίνακας έχει 10 γραμμές πέρα από την επικεφαλίδα και κάθε γραμμή αντιστοιχεί σε ένα επίπεδο. Τον πίνακα ακολουθεί η λεζάντα με τα δεδομένα χρήστη, το συνολικό σκορ και την ημερομηνία ολοκλήρωσης του παιχνιδιού.

Εικόνα 41: Παράδειγμα λεζάντας πίνακα αποτελεσμάτων

Στοιχεία χρήστη --> Ονοματεπώνυμο: Μαρία Μούσιου , e-mail: mm@gmail.com , Score: 210 , Ημερ/νια: 1/5/2021, 10:48 π.μ.

Ο χρήστης έχει τη δυνατότητα να κάνει λήψη των δεδομένων του σε μορφή pdf ή excel.

Πίνακας 8: Απόσπασμα πίνακα αποτελεσμάτων - αρχείο excel

Στοιχεία χρήστη --> Ονοματεπώνυμο: Μαρία Μούσιου , e-mail: mm@gmail.com , Score: 210 , Ημερ/νια: 1/5/2021, 10:48 π.μ.					
Επίπεδο	**Αποτέλεσμα**	**Βαθμολογία**	**Χρόνος**	**Πλήθος play**	**Κώδικας JavaScript**

1	Επιτυχία	10	16:32	2	moveForward();
					turnLeft();
					moveForward();
					moveForward();
					turnRight();
					moveForward();

2	Επιτυχία	20	8:42	2	while (notDone()) {
					turnRight();
					moveForward();
					moveForward();
					moveForward();
					turnLeft();
					moveForward();
					}

3	Επιτυχία	30	6:05	2	for (var count2 = 0; count2 < 3; count2++) { for (var count = 0; count < 4; count++) {

					moveForward();
					}
					turnRight();
					}
					moveForward();

4.6. Ανάλυση του παιχνιδιού με βάση τους στόχους σχεδίασης

Το παιχνίδι ενσωματώνει τις δυο διαστάσεις Υπολογιστικές Έννοιες (Ακολουθίες, Βρόγχοι Επανάληψης, Συνθήκες) και Υπολογιστικές Πρακτικές (Σταδιακή εξέλιξη και επανάληψη βημάτων, Έλεγχος και εντοπισμός σφαλμάτων) των Brennan και Resnick. Στον παρακάτω πίνακα παρουσιάζουμε τη κατανομή των εννοιών και αντιλήψεων του πλαισίου ανά επίπεδο στην πρώτη έκδοση του aMazeD και με τικ σημειώνουμε το επίπεδο στο οποίο ο χρήστης έρχεται σε επαφή με την συγκεκριμένη έννοια/πρακτική με βάση τα μπλοκ που υπάρχουν στον χώρο εργασίας και τις κινήσεις που πρέπει να γίνουν για να βρεθεί ολοκληρωθεί η λύση.

Πίνακας 9: Υπολογιστικές Πρακτικές και Έννοιες ανά επίπεδο

Στοιχείο ΥΣ	**Επίπεδο**									
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Ακολουθίες	✓						✓			
Βρόγχοι Επανάληψης		✓	✓	✓	✓			✓	✓	✓
Συνθήκες				✓	✓	✓				
Σταδιακή εξέλιξη και επανάληψη βημάτων	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Έλεγχος και εντοπισμός σφαλμάτων	✓	✓	✓		✓		✓			

Ειδικά για την πρακτική «Έλεγχος και εντοπισμός σφαλμάτων» θεωρούμε ότι χρησιμοποιείται στα επίπεδα στα οποία ο χρήστης πρέπει να τρέξει το πρόγραμμα που υπάρχει στον χώρο εργασίας για να αντιληφθεί ποια λάθη εμφανίζονται και πως θα πρέπει να τροποποιηθεί ο κώδικας αυτός για να επιτευχθεί η λύση του επιπέδου.

Η Βοήθεια με Υποστήριξη υλοποιείται στην πρώτη έκδοση του aMazeD με 3 τρόπους:

1. **Παροχή ημιτελούς λύσης σε κάθε επίπεδο**
   Με αυτό τον τρόπο ο χρήστης μπορεί να τρέξει τον κώδικα που βλέπει και να δει στην πράξη την χρήση συγκεκριμένων εννοιών Υπολογιστικής Σκέψης.
2. **Παροχή οδηγιών και επεξηγήσεων των υπολογιστικών εννοιών που είναι απαραίτητες για την ολοκλήρωση της λύσης**
   Μηνύματα βοήθειας που αφορούν την λειτουργικότητα των μπλοκ που βλέπει ο χρήστης στην ημιτελή λύση ή των μπλοκ που υπάρχουν στην εργαλειοθήκη του. Για παράδειγμα η πρώτη συμβουλή που εμφανίζεται στο 6ο επίπεδο. Επιπρόσθετα ο χρήστης έχει στην διάθεση του και μηνύματα εργαλείου (tooltips) για κάθε ένα από αυτά τα μπλοκ.
3. **Παροχή βοήθειας σχετικά με την λογική που ικανοποιεί η λύση**
   Οδηγίες και μηνύματα βοήθειας που εξηγούν στον χρήστη τη λογική του παιχνιδιού, για παράδειγμα η δεύτερη συμβουλή που εμφανίζεται στον χρήστη στο 6ο επίπεδο αφού έχει εκτελέσει το πρόγραμμα του με το πλήκτρο «Παίξε» 2 ή περισσότερες φορές.

4.7. Σύνοψη

Το aMazeD είναι ένα εκπαιδευτικό παιχνίδι δέκα επιπέδων που τροποποιεί τον κώδικα των παιχνιδιών Blockly και προσθέτει σε αυτά αρκετές λειτουργικότητες όπως είναι το χρονόμετρο, η βαθμολογία των επιπέδων και η χρήση των πλήκτρων «Παίξε» και «Υποβολή» για τον έλεγχο του προγράμματος και την αποθήκευση της λύσης αντίστοιχα. Το παιχνίδι διατίθεται σε δυο εκδόσεις. Η πρώτη έκδοση κάνει χρήση της Μάθησης με Υποστήριξη με τρεις τρόπους και εφαρμόζει τις διαστάσεις του πλαισίου ΥΣ που αναπτύχθηκε από τους Brennan και Resnick. Η δεύτερη έκδοση έχει τα ίδια προβλήματα και λύσεις ανά επίπεδο με τη διαφορά ότι δεν παρέχει ημιτελείς λύσεις στο χρήστη και κανενός είδους βοήθεια πέρα από τις οδηγίες στην αρχή του επιπέδου.

5. Υλοποίηση και ανάπτυξη του παιχνιδιού aMazeD

5.1. Εισαγωγή

Στο κεφάλαιο αυτό παρουσιάζουμε τον κώδικα του παιχνιδιού aMazeD και τον τρόπο με τον οποίο υλοποιήθηκαν προγραμματιστικά τα χαρακτηριστικά του παιχνιδιού που περιγράψαμε στο προηγούμενο κεφάλαιο. Αρχικά παρουσιάζεται η δομή των αρχείων τόσο των παιχνιδιών Blockly όσο και του aMazeD, έπειτα παρουσιάζεται ο τρόπος με τον οποίο όλα τα αρχεία συνδέονται και οι εξαρτήσεις των αρχείων. Τέλος αναλύονται οι βασικές συναρτήσεις και μεταβλητές της JavaScript και οι λειτουργίες που επιτελούν στο παιχνίδι και παρατίθεται ο κώδικας για την διεπαφή του παιχνιδιού, τα γραφικά, τα μπλοκ και τον τρόπο που λειτουργεί το παιχνίδι. Ο κώδικας του παιχνιδιού aMazeD και στις δυο εκδόσεις του είναι διαθέσιμος στο GitHub στους συνδέσμους

έκδοση 1: https://github.com/mmousiou/aMazeD
έκδοση 2: https://github.com/mmousiou/aMazeD2

Το aMazeD είναι διαθέσιμο στο διαδίκτυο στους συνδέσμους

έκδοση 1: https://mazegamemm.000webhostapp.com/maze.html

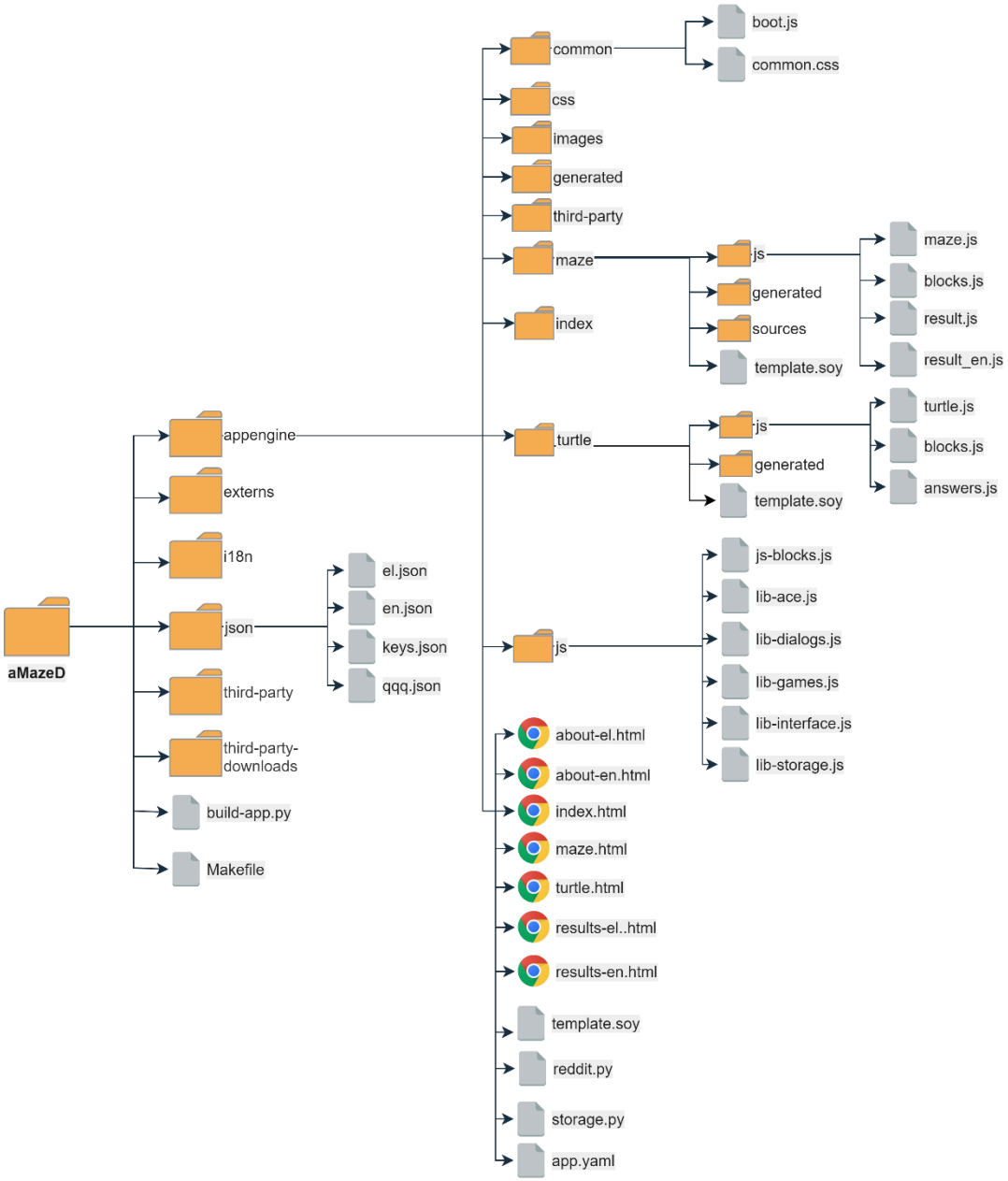
έκδοση 2: https://mazenocomments.000webhostapp.com/maze.html

ενώ οι πληροφορίες για την δημοσίευση του παιχνιδιού είναι διαθέσιμες στο Παράρτημα Γ.

5.2. Δομή αρχείων του παιχνιδιού

Το παιχνίδι aMazeD στηρίζεται στην λογική των παιχνιδιών Blockly, για την ανάπτυξη του έγινε λήψη του κώδικα των παιχνιδιών Blockly από το GitHub και τροποποίηση του κώδικα αυτού. Κάθε παιχνίδι Blockly αποτελείται από ένα φάκελο με το όνομα του παιχνιδιού που περιέχει όλα τα αρχεία που είναι απαραίτητα για την λειτουργία του συγκεκριμένου παιχνιδιού καθώς και κάποια κοινά αρχεία για όλα τα παιχνίδια, τα οποία συνδέονται μεταξύ τους με την χρήση των εργαλείων Closure Tools της Google, ενώ με την βοήθεια του ειδικού αρχείου Makefile δημιουργούνται οι εξαρτήσεις των αρχείων αυτών. Οι φάκελοι, υπό φάκελοι και τα βασικά αρχεία του aMazeD παρουσιάζονται στο Διάγραμμα 2.

Διάγραμμα 2: Δομή φακέλων και αρχείων του παιχνιδιού aMazeD, παρουσιάζονται τα περιεχόμενα των υπο φακέλων json, common, maze, turtle, js στα οποία αναφερόμαστε στην ΔΕ[9]



5.2.1. Βιβλιοθήκη Closure και πρότυπα (Closure Templates)

Το παιχνίδι aMazeD αποτελείται από πολλά αρχεία που αλληλοσυνδέονται με την χρήση των εργαλείων Google Closure Tools και πιο συγκεκριμένα της βιβλιοθήκης Closure και των προτύπων Closure.

[9] Το διάγραμμα έγινε με την χρήση του app.diagrams.net

Η Closure είναι μια βιβλιοθήκη της JavaScript που ανήκει στο σύνολο των εργαλείων ανοιχτού κώδικα Google Closure Tools[10] και χρησιμοποιείται δημιουργία εφαρμογών με μεγάλη έκταση και πολυπλοκότητα όπως Gmail, Google Maps και Google Search.

Τα πρότυπα Closure είναι ένα ακόμη εργαλείο των Closure Tools που χρησιμοποιείται στο παιχνίδι και αναπτύχθηκαν για την δημιουργία δυναμικών ιστοσελίδων με JavaScript. Μπορούν να κάνουν χρήση βρόχων, μεταβλητών και παραμέτρων κάτι το οποίο δεν συμβαίνει με τον αντίστοιχο κώδικα σε HTML.

Κάθε κατηγορία (Λαβύρινθος ή Ζωγραφική) του aMazeD αποτελείται από ένα αρχείο HTML, ένα closure template με όνομα αρχείου template.soy και 2-3 αρχεία js με τον κώδικα της κάθε κατηγορίας. Τα αρχεία αυτά βρίσκονται στους φακέλους maze και turtle του appengine εκτός από τα HTML αρχεία που βρίσκονται όλα μαζί στον φάκελο appengine. Επιπλέον υπάρχουν κοινά αρχεία js που καθορίζουν τις γενικότερες λειτουργίες του παιχνιδιού όπως είναι η επιλογή γλώσσας ή η αλλαγή επιπέδων και περιέχουν τις βασικές συναρτήσεις και μεταβλητές που χρησιμοποιούνται και στα δυο παιχνίδια, τα αρχεία αυτά βρίσκονται στους φακέλους common και js (Διάγραμμα 2).

Με την χρήση των συναρτήσεων goog.provide() και goog.require() της βιβλιοθήκης Closure γίνεται σύνδεση των αρχείων JavaScript και template.soy. Κάθε αρχείο js ξεκινά με τη συνάρτηση goog.provide() όπου σαν όρισμα δέχεται ένα αλφαριθμητικό που αντιστοιχεί σε μια μοναδική περιοχή ονόματος (namespace) με τη συνάρτηση αυτή μπορεί να δημιουργηθεί μια μοναδική σύνδεση του αρχείου με το συγκεκριμένο όνομα. Μετά τον ορισμό της μοναδικής εξάρτησης του αρχείου με το όνομα, έχουμε την δυνατότητα να συνδέσουμε το αρχείο με άλλα αρχεία JavaScript μέσω της συνάρτησης goog.require('namespace'). Η goog.require('namespace') δημιουργεί τις εξαρτήσεις του αρχείου αυτού με άλλα αρχεία js στα οποία έχει αντιστοιχιστεί η περιοχή ονόματος που είναι παράμετρος της συνάρτησης.

[10] https://developers.google.com/closure - Closure Tools

Κώδικας 1: Στιγμιότυπο από τον κώδικα του αρχείου maze.js

```
11  'use strict';
12
13  goog.provide('Maze');
14
15  goog.require('Blockly.FieldDropdown');
16  goog.require('Blockly.Trashcan');
17  goog.require('Blockly.utils.dom');
18  goog.require('Blockly.utils.math');
19  goog.require('Blockly.utils.string');
20  goog.require('Blockly.VerticalFlyout');
21  goog.require('BlocklyDialogs');
22  goog.require('BlocklyGames');
23  goog.require('BlocklyInterface');
24  goog.require('Maze.Blocks');
25  goog.require('Maze.soy');
```

Στο στιγμιότυπο Κώδικας 1 βλέπουμε την χρήση της βιβλιοθήκης Closure στο αρχείο JavaScript maze.js . Στην γραμμή 11 γίνεται η χρήση της έκφρασης ‘use strict’ , με την οποία υποδεικνύουμε ότι ο κώδικας JavaScript του αρχείου θα αναπτυχθεί σε αυστηρό περιβάλλον (strict mode), στο οποίο για παράδειγμα δεν μπορούμε να χρησιμοποιήσουμε μεταβλητές που δεν έχουν δηλωθεί εκ των προτέρων. Στην γραμμή 13 με την χρήση της goog.provide(‘Maze’) γίνεται η σύνδεση του αρχείου με την περιοχή ονόματος ‘Maze’, έτσι κάθε αρχείο που θα έχει στην αρχή την εντολή goog.require(‘Maze’) θα μπορεί να χρησιμοποιεί τον κώδικα του αρχείου maze.js. Στις γραμμές 15-25 μπορούμε να δούμε τις περιοχές ονομάτων δηλαδή τα αρχεία js με τα οποία είναι συνδεδεμένα το maze.js. Για παράδειγμα στη γραμμή 24 γίνεται η σύνδεση του maze.js με το Maze.Blocks που είναι ο χώρος ονόματος που έχει κατοχυρωθεί από το αρχείο blocks.js με διαδρομή appengine/maze/js/blocks.js.

Η διεπαφή του παιχνιδιού aMazeD ορίζεται από τρία template.soy αρχεία, ένα γενικό στον υπο φάκελο appengine που καθορίζει γενικά χαρακτηριστικά όπως η γραμμή πλοήγησης και τα άλλα δυο στους υπο φακέλους maze και turtle (Διάγραμμα 2), που προσδιορίζουν πιο συγκεκριμένα χαρακτηριστικά των επιπέδων 1-6 και 7-10 αντίστοιχα. Τα αρχεία αυτά είναι πρότυπα Closure (closure template). Τα πρότυπα Closure μπορούν να ομαδοποιηθούν και να καλούν άλλα πρότυπα Closure, έτσι για παράδειγμα στο πρότυπο template.soy του maze καλούνται τα templates που έχουν οριστεί στον πρότυπο template.soy του appengine. Επιπλέον τα πρότυπα Closure μπορούν να καλούνται από αρχεία js. Στο στιγμιότυπο Κώδικας 1 στη γραμμή 25 γίνεται κλήση του προτύπου template.soy του φακέλου maze, το οποίο έχει αντιστοιχιστεί στο όνομα maze.soy. Με αυτόν τον τρόπο όλα τα στοιχεία div, classes και όλα τα μηνύματα που περιέχονται στο πρότυπο με ευρετήριο

appengine/maze/template.soy γίνονται διαθέσιμα στο αρχείο maze.js και μπορούν να τροποποιηθούν ή να συνδεθούν με συμβάντα (events).

Κώδικας 2: Στιγμιότυπο του appengine/maze/template.soy

```
1   {namespace Maze.soy}
2
3   /**
4    * This is a Closure Template.
5    *
6    * To regenerate just English, run:
7    *   make maze-en
8    *
9    * To regenerate all languages, run:
10   *   make languages
11   */
12
13  /**
14   * Translated messages for use in JavaScript.
15   */
16  {template .messages}
17    {call BlocklyGames.soy.messages /}
```

Στο στιγμιότυπο Κώδικας 2 βλέπουμε τις πρώτες γραμμές κώδικα του προτύπου template.soy του παιχνιδιού Λαβύρινθος, με ευρετήριο appengine/maze/template.soy. Στην πρώτη γραμμή γίνεται η σύνδεση του αρχείου με το χώρο ονόματος maze.soy. Στην γραμμή 17 καλείται μέσα στο πρότυπο maze.soy το πρότυπο «μηνύματα» (.messages) που υπάρχει στο σώμα του BlocklyGames.soy που είναι ο χώρος ονόματος που έχει αντιστοιχιστεί στο Closure πρότυπο με ευρετήριο appengine/template.soy.

5.2.2. Εκτέλεση του παιχνιδιού σε περιβάλλον τοπικού εξυπηρετητή

Το πιο διαδεδομένο πρότυπο για την δημιουργία ενός παιχνιδιού είναι κάθε επίπεδο να έχει τη δική του δυναμική ιστοσελίδα HTML και συνδέεται με τα αρχεία js που έχουν την λειτουργικότητα του παιχνιδιού. Το aMazeD έχει τελείως διαφορετική δομή. Στο παιχνίδι υπάρχουν 5 σελίδες HTML οι about.html, results.html που έχουμε αναφέρει και στο κεφάλαιο 4 και οι index.html, maze.html και turtle.html, που έχουν υιοθετηθεί από τα Παιχνίδια Blockly και χρησιμοποιούνται για την μετάβαση του φυλλομετρητή στο αντίστοιχο παιχνίδι. Στο παιχνίδι aMazeD η index.html και maze.html οδηγούν στο πρώτο επίπεδο του παιχνιδιού ενώ η turtle.html στο 7^{ο} επίπεδο του παιχνιδιού. Αν δοκιμάσουμε να τρέξουμε τοπικά κάποια από τις ιστοσελίδες maze.html , index.html, turtle.html είτε με διπλό κλικ είτε μέσω Xampp θα δούμε μια κενή σελίδα στο φυλλομετρητή καθώς τα αρχεία

HTML του aMazeD δεν περιέχουν κώδικα για την δομή της ιστοσελίδας του παιχνιδιού, όπως ήδη έχει αναφερθεί η δομή γίνεται με το template.soy. Τα αρχεία αυτά αποτελούν απλά έναν οδηγό για την σύνδεση του φυλλομετρητή στο παιχνίδι aMazeD. Ο κώδικας είναι παρόμοιος και στα 3 αυτά αρχεία και αποτελείται από την σύνδεση της σελίδας μέσω της ετικέτας link με το αντίστοιχο αρχείο μορφοποίησης (με κατάληξη .css[11]) για την κάθε κατηγορία (maze ή turtle), με το κοινό αρχείο μορφοποίησης common.css και με το κοινό αρχείο boot.js για όλα τα παιχνίδια.

Κώδικας 3: Κώδικας HTML για τον Λαβύρινθο

```
<!DOCTYPE html>
<html>

<head>
    <meta charset="utf-8" />
    <meta name="google" value="notranslate" />
    <meta name="viewport" content="target-densitydpi=device-dpi, width=device-width, initial-scale=1.0,
    user-scalable=no" />
    <meta name="description" content="Levels 1-6 are based on Maze. In each level player must use the
    blocks to create a programm in order to move the character to the finish point." />
    <title>aMazeD : Maze</title>
    <link rel="stylesheet" href="common/common.css" />
    <link rel="stylesheet" href="maze/style.css" />
    <script src="common/boot.js"></script>
</head>

<body>
    <noscript>aMazeD requires JavaScript.</noscript>
</body>

</html>
```

Ο κώδικας των παιχνιδιών Blockly και κατ' επέκταση του aMazeD είναι συμπιεσμένος και βρίσκεται στους φακέλους generated. Αρχικά πρέπει να γίνει δόμηση του παιχνιδιού στην γραμμή εντολών του λειτουργικού συστήματος[12] (cmd - Windows command line) πριν μπορέσουμε να τρέξουμε το παιχνίδι τοπικά. Οι εντολές είναι τρεις και γίνονται με την χρήση ψευδοκώδικα Linux.

- Η πρώτη εντολή που πρέπει να εκτελεστεί είναι η make deps η οποία δημιουργεί τις κατάλληλες εξαρτήσεις μεταξύ των αρχείων JavaScript, των προτύπων Closure και

[11] cascading style sheet

[12] Το λειτουργικό σύστημα που χρησιμοποιήθηκε είναι των Windows ωστόσο ο κώδικας των Παιχνιδιών Blockly προορίζεται για χρήση σε Unix τύπου λειτουργικά συστήματα όπως είναι τα Linux και Mac OS. Για το λόγο αυτό και οι εντολές δεν μπορούν να εκτελεστούν απευθείας στην γραμμή εντολών των Windows αλλά πρέπει να χρησιμοποιηθεί πρώτα ένα υποσύστημα τύπου Linux.

των αρχείων της βιβλιοθήκης Blockly και των αρχείων python που περιέχουν τις εντολές για την συμπίεση και αποσυμπίεση του κώδικα.

Κώδικας 4: Εντολή make deps στην γραμμή εντολών των Windows

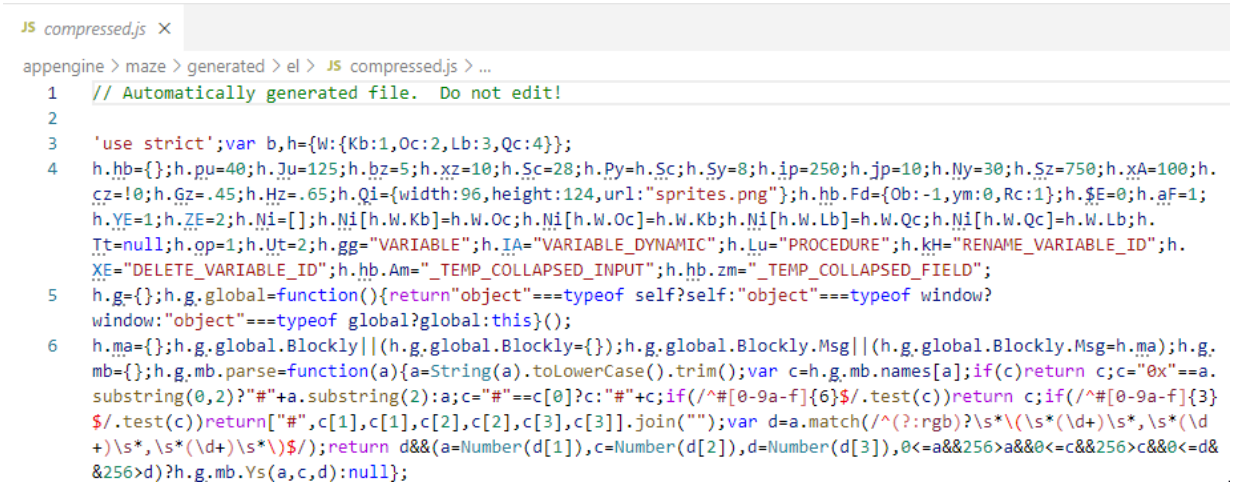
```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\[redacted]> cd .\Desktop\
PS C:\Users\[redacted]\Desktop> cd .\aMazeD-main\
PS C:\Users\[redacted]\Desktop\aMazeD-main> wsl
[redacted]:/mnt/c/Users/[redacted]/Desktop/aMazeD-main$ make deps
Found `svn`
Found `unzip`
Found `wget`
Found `java`
Found `python`
Found `sed`
mkdir -p third-party-downloads
cd third-party-downloads; \
svn checkout https://github.com/google/closure-library/trunk/closure/bin/build build; \
wget -N https://dl.google.com/closure-templates/closure-templates-for-javascript-latest.zip; \
unzip -o closure-templates-for-javascript-latest.zip SoyToJsSrcCompiler.jar; \
wget -N https://dl.google.com/closure-templates/closure-templates-msg-extractor-latest.zip; \
unzip -o closure-templates-msg-extractor-latest.zip SoyMsgExtractor.jar; \
wget -N https://unpkg.com/google-closure-compiler-java/compiler.jar; \
mv -f compiler.jar closure-compiler.jar; \
chmod +x build/closurebuilder.py
A    build/depswriter.py
A    build/source.py
A    build/treescan.py
Checked out revision 11473.
```

Την εκτελούμε μόνο την πρώτη φορά που κάνουμε λήψη του παιχνιδιού.

- Οι άλλες δυο εντολές είναι οι make maze και make turtle που κάνουν δόμηση στα επίπεδα του παιχνιδιού Λαβύρινθος και Ζωγραφική στα ελληνικά και στα αγγλικά. Όταν εκτελεστούν την πρώτη φορά θα μπορούμε να τρέξουμε το παιχνίδι τοπικά ωστόσο αυτές οι δυο εντολές πρέπει εκτελούνται κάθε φορά που θέλουμε να κάνουμε κάποια αλλαγή στα αρχεία js ή template.soy του παιχνιδιού και θέλουμε να ενσωματώσουμε τις αλλαγές αυτές στο παιχνίδι.

Με τις τελευταίες δυο εντολές συμπιέζονται όλα τα αρχεία των maze και turtle και κοινά αρχεία του παιχνιδιού και δημιουργούνται τα αρχεία min.js που μας επιτρέπουν να τρέξουμε το aMazeD σε περιβάλλον τοπικού εξυπηρετητή. Τα κοινά αρχεία min.js αποθηκεύονται στο φάκελο generated του appengine ενώ τα αρχεία για τον Λαβύρινθο και την Ζωγραφική στους υπο φακέλους generated των φακέλων maze και turtle. Επιπλέον ενημερώνονται οι εξαρτήσεις με τα αρχεία .json που περιέχουν τα μηνύματα του παιχνιδιού και στις δυο γλώσσες με τα αρχεία template.soy και js του παιχνιδιού. Έτσι τα αρχεία min.js κατηγοριοποιούνται με βάση την γλώσσα σε δυο φακέλους el, en. Εδώ γίνεται αντιληπτό ότι στην περίπτωση των παιχνιδιών Blockly η συμπίεση και αποσυμπίεση γίνεται σε 50 υπο φακέλους που αντιστοιχούν στις διαφορετικές γλώσσες που είναι διαθέσιμα τα παιχνίδια.

Εικόνα 42: Απόσπασμα από το συμπιεσμένο αρχείο για τον Λαβύρινθο στα ελληνικά



Το σύνολο των κανόνων με τους οποίους οι εντολές make deps, make maze και make turtle εκτελούν τις εντολές που περιγράφονται στα αρχεία python και δημιουργούν τις εξαρτήσεις, βρίσκονται στο αρχείο Makefile. Το αρχείο Makefile είναι ένα ειδικό αρχείο που λέει στην εντολή make τι να κάνει, πως να μεταφράσει και πως να συνδέσει το πρόγραμμα (An Introduction to Makefiles, MIT). Στο απόσπασμα Κώδικας 5, στην γραμμή 5 έχουμε τις εφαρμογές στις οποίες θα δημιουργηθούν οι εξαρτήσεις, στις γραμμές 6-11 αναφέρονται τα πρότυπα Closure που θα διαβαστούν από τον μεταφραστή και ο τρόπος με τον οποίο θα γίνει η μεταγλώττισή τους, στην γραμμή 13 αναφέρονται τα απαιτούμενα προγράμματα που αν δεν υπάρχουν στο λειτουργικό σύστημα θα εγκατασταθούν, τέλος οι γραμμές 41-46 αφορούν τις εντολές make maze, make turtle και τα συμπιεσμένα αρχεία που θα δημιουργηθούν μαζί με τους φακέλους generated.

Κώδικας 5: Απόσπασμα κώδικα από το αρχείο Makefile

Makefile

```
1  ################################
2  # Definitions
3  ################################
4
5  USER_APPS = {index,maze,turtle}
6  ALL_JSON = {./,index,maze,turtle}
7  ALL_TEMPLATES = appengine/template.soy,appengine/index/template.soy,
   appengine/maze/template.soy,appengine/turtle/template.soy
8
9  APP_ENGINE_THIRD_PARTY = appengine/third-party
10 SOY_COMPILER = java -jar third-party-downloads/SoyToJsSrcCompiler.jar
   --shouldProvideRequireSoyNamespaces --isUsingIjData
11 SOY_EXTRACTOR = java -jar third-party-downloads/SoyMsgExtractor.jar
12
13 REQUIRED_BINS = svn unzip wget java python sed
14
```

```
41  index  maze turtle : common
42      @echo "Generating JS from appengine/$@/template.soy"
43      mkdir -p appengine/$@/generated;
44      i18n/json_to_js.py --output_dir appengine/$@/generated --template
        appengine/$@/template.soy json/*.json;
45      python build-app.py $@
46      @echo
```

5.2.3. Κοινά αρχεία, μεταβλητές και συναρτήσεις

Ο κώδικας του παιχνιδιού όπως είπαμε μοιράζεται σε πολλά js αρχεία και 3 αρχεία προτύπων Closure. Μια από τις βασικές αρχές της υλοποίησης του παιχνιδιού είναι ότι χρησιμοποιούνται κάποιες σταθερές και συναρτήσεις που είναι κοινές σε όλα τα αρχεία του παιχνιδιού είτε ανήκουν στον Λαβύρινθο είτε ανήκουν στη Χελώνα. Επειδή ο κώδικας των αρχείων κληρονομήθηκε από τα Παιχνίδια Blockly κρατήθηκαν τα αρχικά ονόματα αυτών των μεταβλητών – συναρτήσεων. Για παράδειγμα οι σταθερές που υπάρχουν στο αρχείο lib-games.js έχουν πρόθεμα BlocklyGames.ΣΤΑΘΕΡΑ, όπως η σταθερά BlocklyGames.LEVEL στην οποία αποθηκεύεται το τρέχον επίπεδο του χρήστη. Παρόμοια είναι και η ονομασία των συναρτήσεων, οι συναρτήσεις για παράδειγμα του lib-interface έχουν πρόθεμα BlocklyInterface.όνομα_συνάρτησης(παράμετροι) όπως είναι η BlocklyInterface.eventSpam(). Οι συναρτήσεις του Λαβυρίνθου είναι της μορφής Maze.όνομα_συνάρτησης(παράμετροι) και οι συναρτήσεις της Ζωγραφικής είναι της μορφής Turtle.όνομα_συνάρτησης(παράμετροι).

5.3. Διεπαφή του παιχνιδιού

5.3.1. Επίπεδα

Το παιχνίδι έχει 10 επίπεδα και ο καθορισμός του μέγιστου αριθμού επιπέδων γίνεται με την σταθερά BlocklyGames.MAX_LEVEL η οποία έχει σαν τιμή τον μέγιστο αριθμό επιπέδων και για το aMazeD είναι ίση με 10. Για να μπορούμε να ελέγχουμε το επίπεδο στο οποίο βρίσκεται ο χρήστης χρησιμοποιούμε την μεταβλητή

```
BlocklyGames.LEVEL =
    BlocklyGames.getNumberParamFromUrl(
        'level', 1, BlocklyGames.MAX_LEVEL);
```

Στην μεταβλητή BlocklyGames.LEVEL αποθηκεύεται ο αριθμός του επιπέδου που θα επιστρέψει η συνάρτηση BlocklyGames.getNumberParamFromUrl. Η τελευταία θα ψάξει στο URL της ιστοσελίδας την ακέραια τιμή του 'level' στο διάστημα 1-10 (δεδομένου ότι στο παιχνίδι aMazeD φτάνουμε μέχρι το επίπεδο 10) και θα επιστρέψει τον αριθμό του επιπέδου. Στην Εικόνα 43 βλέπουμε τις πληροφορίες που μπορούμε να πάρουμε από το URL του παιχνιδιού και συγκεκριμένα του Λαβυρίνθου

- maze.html είναι η ιστοσελίδα που μας δείχνει ότι βρισκόμαστε στον Λαβύρινθο
- lang=el η γλώσσα του παιχνιδιού είναι Ελληνικά
- level=2 βρισκόμαστε στο 2ο επίπεδο
- skin=0 η εμφάνιση επιπέδου είναι ο αστροναύτης

Εικόνα 43: URL επιπέδου 2 του παιχνιδιού σε τοπικό εξυπηρετητή

Για να μεταβούμε από το ένα επίπεδο στο άλλο χρησιμοποιούμε την συνάρτηση BlocklyInterface.nextLevel του αρχείου lib-interface.js όπως φαίνεται στον Κώδικα 6.

Κώδικας 6: Συνάρτηση αλλαγής επιπέδου στο lib-interface.js

```
256  BlocklyInterface.nextLevel = function() {
257      if (BlocklyGames.LEVEL < BlocklyGames.MAX_LEVEL) {
258          window.location = window.location.protocol + '//' +
259              window.location.host + window.location.pathname +
260              '?lang=' + BlocklyGames.LANG + '&level=' + (BlocklyGames.LEVEL + 1);
261      } else {
262          BlocklyInterface.indexPage();
263      }
264  };
```

Στην γραμμή 257 βρίσκεται η συνθήκη ελέγχου της συνάρτησης, αν το τρέχον επίπεδο είναι μικρότερο από το μέγιστο επίπεδο που έχει οριστεί τότε το νέο URL θα αυξήσει κατά μια μονάδα την τιμή του 'level', αν η συνθήκη είναι ψευδής θα εκτελεστεί η συνάρτηση BlocklyInterface.indexPage(), η οποία περιγράφεται στο ίδιο αρχείο και ανακατευθύνει τον χρήστη στην σελίδα των αποτελεσμάτων με την εντολή

```
window.location = 'results-' + BlocklyGames.LANG + '.html';
```

που βρίσκεται στο σώμα της συνάρτησης BlocklyInterface.indexPage().

Το window.location είναι ένα αντικείμενο της JavaScript με το οποίο μπορούμε να πάρουμε το τρέχον URL και να ανακατευθύνουμε τον φυλλομετρητή σε μια νέα σελίδα[13]. Η

[13] https://www.w3schools.com/js/js_window_location.asp

μεταβλητή BlocklyGames.LANG είναι τύπου αλφαριθμητικό και έχει δυο πιθανές επιλογές en, el για να υπάρχει αντιστοιχία της επιθυμητής γλώσσας χρήστη με την γλώσσα στη σελίδα των αποτελεσμάτων[14].

Η συνάρτηση BlocklyInterface.nextLevel επικαλύπτεται στην περίπτωση του Λαβυρίνθου. Ο Λαβύρινθος αποτελεί τα επίπεδα 1-6 του aMazeD, ως αποτέλεσμα η συνάρτηση όπως ορίστηκε παραπάνω δεν θα καταφέρει να ανακατευθύνει τον χρήστη στο επίπεδο 7, έτσι τροποποιήσαμε τη συνάρτηση στο αρχείο maze.js όπως φαίνεται στο στιγμιότυπο Κώδικας 7.

Κώδικας 7: Επικάλυψη της συνάρτησης αλλαγής επιπέδου στο maze.js

```
39  BlocklyInterface.nextLevel = function() {
40      if (BlocklyGames.LEVEL < 6) {
41          window.location = window.location.protocol + '//' +
42              window.location.host + window.location.pathname +
43              '?lang=' + BlocklyGames.LANG + '&level=' + (BlocklyGames.LEVEL + 1) +
44              '&skin=' + Maze.SKIN_ID;
45      } else if (BlocklyGames.LEVEL == 6) {
46          BlocklyInterface.turtlePage();
47      }
48  };
```

Η συνθήκη της συνάρτησης if στη γραμμή 40 ελέγχει αν το τρέχον επίπεδο είναι μικρότερο από το 6 και αν είναι αληθής η τιμή του ορίσματος level αυξάνεται κατά ένα έτσι όταν θα ξαναδιαβαστεί το επίπεδο θα είναι αυξημένο κατά ένα, ενώ αν είναι ψευδής καλεί την συνάρτηση BlocklyInterface.turtlePage() η οποία ανακατευθύνει το χρήστη στο turtle.html με την χρήση του κώδικα

```
window.location = 'turtle.html' + '?lang=' + BlocklyGames.LANG;
```

στο σώμα της συνάρτησης.

Το παιχνίδι aMazeD χρησιμοποιεί σε πάρα πολλές περιπτώσεις το LocalStorage του φυλλομετρητή. Το LocalStorage είναι ακόμα μια ιδιότητα του φυλλομετρητή και δίνει τη δυνατότητα να αποθηκεύουμε τοπικά στον φυλλομετρητή ζεύγη αντικειμένων τύπου όνομα-τιμή, όπου το όνομα είναι υποχρεωτικά αλφαριθμητικό ενώ η τιμή μπορεί να είναι αλφαριθμητικό ή αριθμός.

Στο aMazeD με την συνάρτηση BlocklyInterface.saveToLocalStorage() αποθηκεύουμε τοπικά το επίπεδο του χρήστη σαν όνομα (για παράδειγμα το επίπεδο 1 του Λαβυρίνθου θα αποθηκευτεί ως maze1, το επίπεδο 8 της ζωγραφικής ως turtle8) και σαν τιμή δίνουμε τα

[14] results-en.html είναι η αγγλική έκδοση της σελίδας, ενώ results-en.html είναι η ελληνική έκδοση της σελίδας.

μπλοκ που χρησιμοποίησε ο χρήστης στο επίπεδο αυτό με την μορφή xml κειμένου. Με την συνάρτηση BlocklyGames.loadFromLocalStorage γίνεται ανάκτηση των πληροφοριών από τον φυλλομετρητή.

Κώδικας 8: Κώδικας για έλεγχο αποθηκευμένων επιπέδων και τρέχον επίπεδο

```
111  for (var i = 0; i < 6; i++) {
112      if (!BlocklyGames.loadFromLocalStorage(BlocklyGames.NAME, i + 1))
         {
113          BlocklyGames.LEVEL = i + 1;
114          break;
115      } else if (!!BlocklyGames.loadFromLocalStorage(BlocklyGames.NAME,
         6)) {
116          BlocklyGames.LEVEL = 6;
117      }
118  }
```

Στον Κώδικα 8 του maze.js παρουσιάζουμε τον κώδικα με τον οποίο εκχωρούμε το τρέχον επίπεδο στη μεταβλητή BlocklyGames.Level. Ξεκινώντας από το επίπεδο 1 του aMazeD ($i = 0$) ελέγχουμε αν το επίπεδο βρίσκεται στο LocalStorage, αν δεν βρίσκεται (δηλαδή η συνθήκη στη γραμμή 12 είναι αληθής) τότε το τρέχον επίπεδο είναι το 1 και ο βρόγχος τερματίζει, αν το επίπεδο 1 βρίσκεται στο LocalStorage , δηλαδή ο χρήστης έχει παίξει το επίπεδο αυτό, δεν εκτελούνται οι εντολές μέσα στο σώμα του if και ο μετρητής του for αυξάνει κατά 1 μέχρι να βρεθεί ένα επίπεδο που να μην είναι στο LocalStorage ή μέχρι να φτάσουμε στο 6ο επίπεδο που είναι το τελευταίο του Λαβυρίνθου. Έπειτα στην συνάρτηση Maze.init() που εκτελείται κάθε φορά που ανανεώνεται η σελίδα maze.html αρχικοποιούμε τις τιμές των ιδιοτήτων lang, level, maxLevel, skin, html του προτύπου Closure maze.soy και BlocklyGames.soy.

Κώδικας 9: Αρχικοποίηση παραμέτρων προτύπων Closure

```
492  Maze.init = function() {
493
494      // Render the Soy template.
495      document.body.innerHTML = Maze.soy.start({}, null, {
496          lang: BlocklyGames.LANG,
497          level: BlocklyGames.LEVEL,
498          maxLevel: 10,
499          skin: Maze.SKIN_ID,
500          html: BlocklyGames.IS_HTML
501      });
```

Αντίστοιχα για να ξεκινήσουμε από το επίπεδο 7 στη Ζωγραφική χρησιμοποιούμε τον κώδικα 10. Στις γραμμές 97-101 με την χρήση του for τοποθετούμε τα επίπεδα 1-6 του turtle στο LocalStorage χρησιμοποιώντας το

```
window.localStorage['όνομα'] = κλειδί
```

Στις γραμμές 105-112 χρησιμοποιούμε ένα δεύτερο for βρόγχο που ακολουθεί την λογική του κώδικα 8 του maze.js. Αν η συνθήκη στη γραμμή 106 είναι αληθής, δηλαδή το επίπεδο 7 δεν βρίσκεται στο LocalStorage του φυλλομετρητή, τότε το τρέχον επίπεδο είναι το 7, αλλιώς επαναλαμβάνεται ο έλεγχος μέχρι να φτάσουμε στο επίπεδο 10.

Κώδικας 10: Στιγμιότυπο από turtle.js για επιλογή επιπέδου σε Ζωγραφική (επίπεδα 7-10)

```
97    for (var i = 1; i < 7; i++) {
98        BlocklyGames.LEVEL = i;
99        var tname = BlocklyGames.NAME + BlocklyGames.LEVEL;
100       window.localStorage[tname] = "no game";
101   }

105   for (var i = 6; i < BlocklyGames.MAX_LEVEL; i++) {
106       if (!BlocklyGames.loadFromLocalStorage(BlocklyGames.NAME,
          i + 1)) {
107           BlocklyGames.LEVEL = i + 1;
108           break;
109       } else if (!!BlocklyGames.loadFromLocalStorage
          (BlocklyGames.NAME, BlocklyGames.MAX_LEVEL)) {
110           BlocklyGames.LEVEL = BlocklyGames.MAX_LEVEL;
111       }
112   }
```

Με αυτές τις συναρτήσεις αν υπάρχει παιχνίδι σε εξέλιξη ο παίκτης ξεκινά από το επίπεδο που σταμάτησε, ενώ αν έχει ξεκινήσει νέο παιχνίδι ανακατευθύνεται αυτόματα από το ένα επίπεδο στο άλλο, μετά το επίπεδο 6 ανακατευθύνεται στα επίπεδα της Ζωγραφικής και μετά το επίπεδο 10 στη σελίδα των αποτελεσμάτων.

Τα επίπεδα στην διεπαφή του χρήστη βρίσκονται στην γραμμή πλοήγησης δίπλα από το λογότυπο του παιχνιδιού. Ο κώδικας για την δημιουργία της γραμμής πλοήγησης και των στοιχείων της βρίσκεται στο BlocklyGames.soy (ευρετήριο appengine/template.soy) στο πρότυπο headerBar. Με την εντολή call καλούμε πρότυπα που είτε έχουν οριστεί στο συγκεκριμένο πρότυπο Closure είτε σε άλλα πρότυπα Closure. Στη γραμμή 91(Κώδικας 11) για παράδειγμα καλούμε το πρότυπο titleSpan που ορίζεται πιο κάτω στο ίδιο αρχείο, ενώ στη γραμμή 94 καλούμε το πρότυπο levelLinks που ορίζεται στο ίδιο αρχείο στις γραμμές 142-160 και το οποίο είναι υπεύθυνο για την εμφάνιση των επιπέδων στην γραμμή πλοήγησης.

Κώδικας 11: Στιγμιότυπο προτύπου headerBar στο BlocklyGames.soy

```
{template .headerBar private="true"}
  <table width="100%">
    <tr>
      <td id="header_title" class="leftspan">
        {call .titleSpan}
        {/call}
        {if $ij.level}
          {call .levelLinks}
            {param suffix}{if $levelLinkSuffix}
            {$levelLinkSuffix}{/if}{/param}
          {/call}
        {/if}
      </td>
      <td id="header_cta" class="farSide">
        <select id="languageMenu"></select>
        {if $hasLinkButton}
           
          <button id="linkButton"
                  title="{msg meaning="Games.linkTooltip"
                  desc="IBID"}Save and link to blocks.{/msg}">
            <img src="common/1x1.gif" class="link icon21">
          </button>
        {/if}
        {if $hasHelpButton}
           
          <button id="helpButton">
            {{msg meaning="Games.help" desc="IBID"}}Help{{/msg}
            }
          </button>
        {/if}
        {if $farLeftHtml}
           
          {$farLeftHtml |noAutoescape}
        {/if}
      </td>
    </tr>
  </table>
{/template}
```

Στο πρότυπο headerBar ορίζονται επίσης το μενού επιλογής γλώσσας, το κουμπί βοήθειας που βρίσκεται στα επίπεδα 7-10 και όποια άλλη πληροφορία με το $farLeftHtml. Με την τελευταία παράμετρο εμφανίζουμε στα επίπεδα του Λαβυρίνθου το μενού επιλογής «εμφάνιση επιπέδου». Τα πρότυπα Closure που ελέγχουν τις κατηγορίες Λαβυρίνθου και Ζωγραφικής μπορούν να καλέσουν το .headerBar και να προσδιορίσουν τις μεταβλητές του προτύπου όπως φαίνεται στο στιγμιότυπο Κώδικας 12. Στο στιγμιότυπο αυτό βλέπουμε την

κλήση του headerBar στο Maze.soy στην γραμμή 47, τον προσδιορισμό της παραμέτρου appName ως Maze (γραμμή 49), την αντιστοίχιση του skin δηλαδή της εμφάνισης επιπέδου (γραμμή 51) και την δημιουργία του μενού επιλογής «εμφάνιση επιπέδου» στις γραμμές 54-59, μόνο για τα επίπεδα του Λαβυρίνθου.

Κώδικας 12: Πρότυπο start του Maze.soy

```
45  {template .start}
46    {call .messages /}
47    {call BlocklyGames.soy.headerBar}
48      {param appName}
49        {msg meaning="Games.maze" desc="IBID"}Maze{/msg}
50      {/param}
51      {param levelLinkSuffix}skin={$ij.skin}{/param}
52      {param hasLinkButton: true /}
53      {param hasHelpButton: false /}
54      {param farLeftHtml}
55        <button id="pegmanButton">
56          <img src="common/1x1.gif">
57          <span id="pegmanButtonArrow"></span>
58        </button>
59      {/param}
60    {/call}
```

Αντίστοιχα στο πρότυπο start του προτύπου Closure της Ζωγραφικής η παράμετρος hasHelpButton παίρνει τιμή true για να εμφανιστεί το κουμπί βοήθειας που υπάρχει σε αυτά τα επίπεδα ενώ οι παράμετροι levelLinkSuffix και farLeftHtml παραμένουν κενές καθώς δεν υπάρχουν επιπλέον ιδιότητες των επιπέδων όπως είναι η επιλογή εμφάνισης.

Κώδικας 13: Πρότυπο start του Turtle.soy

```
62  {template .start}
63    {call .messages /}
64    {call BlocklyGames.soy.headerBar}
65      {param appName}
66        {msg meaning="Games.turtle" desc="IBID"}Draw{/msg}
67      {/param}
68      {param levelLinkSuffix}{/param}
69      {param hasLinkButton: true /}
70      {param hasHelpButton: true /}
71      {param farLeftHtml}{/param}
72    {/call}
```

Με το πρότυπο titleSpan εμφανίζουμε το λογότυπο του παιχνιδιού. Το λογότυπο αποτελεί υπερσύνδεση για την ιστοσελίδα των πληροφοριών και η σύνδεση αυτή γίνεται μέσω της ετικέτας <a href=’ ‘ > στην γραμμή 130. Η ιστοσελίδα about.html είναι διαθέσιμη σε δυο γλώσσες με το πρόθεμα της γλώσσας στο τίτλο της ιστοσελίδας, έτσι η about-en.html θα

ανακατευθύνει τον χρήστη στην σελίδα στα ελληνικά ενώ η about-en.html στα αγγλικά. Η συνθήκη επιλογής γίνεται και πάλι από την επιλεγμένη γλώσσα και με την βοήθεια της παραμέτρου $ij.lang που θα πάρει σαν τιμή την γλώσσα των επιπέδων.

Κώδικας 14: Πρότυπο titleSpan του BlocklyGames.soy

```
126  {template .titleSpan private="true"}
127    <span id="title">
128      {if $ij.html}
129        <a href="about-{$ij.lang}.html">
130      {else}
131        <a href="./?lang={$ij.lang}">
132      {/if}
133        <img src="images/logo_amazed.png" width="125">
134      </a>
135    </span>
136  {/template}
```

Με το πρότυπο levelLinks αλλάξαμε μια από τις βασικές επιλογές των Παιχνιδιών Blockly. Στα Παιχνίδια Blockly ο παίκτης έχει την δυνατότητα μετάβασης σε όποιο από τα επίπεδα επιθυμεί ακόμα και αν το έχει ολοκληρώσει και με αχνό γκρι χρώμα απλά επισημαίνονται τα ολοκληρωμένα επίπεδα. Στο aMazeD δεν δίνεται αυτή η δυνατότητα. Στην γραμμή 143 βάζουμε κενό (non-breakable space) ανάμεσα στους αριθμούς των επιπέδων, στην γραμμή 146 διαβάζουμε από το url το τρέχον επίπεδο και την γλώσσα του χρήστη, ενώ στο μπλοκ if-else των γραμμών 152-158 αν το επίπεδο είναι ολοκληρωμένο χρωματίζεται αναλόγως με την χρήση της κλάσης `level_number level_done` και των αντίστοιχων ιδιοτήτων της στο common.css αρχείο ενώ οποιαδήποτε επιλογή με το ποντίκι άλλου επιπέδου θα οδηγεί πάντα στο τρέχον επίπεδο.

Κώδικας 15: Πρότυπο levelLinks του BlocklyGames.soy

```
{template .levelLinks private="true"}
  {sp} {sp}
  {for $i in range(1, $ij.maxLevel + 1)}
    {let $url}
      ?lang={$ij.lang}&level={$i}
      {if $suffix}
        &{$suffix}
      {/if}
    {/let}
    {sp}
    {if $i == $ij.level}
      <span class="level_number level_done" id="level{$i}
      ">{$i}</span>
    {elseif $i == $ij.maxLevel}
      <a class="level_number" id="level{$i}">{$i}</a>
    {else}
      <a class="level_number" id="level{$i}">{$i}</a>
    {/if}
  {/for}
{/template}
```

5.3.2. Επιλογή γλώσσας

Στο παιχνίδι aMazeD υπάρχουν δυο διαθέσιμες γλώσσες τα Ελληνικά, που είναι η προεπιλεγμένη γλώσσα και τα Αγγλικά. Οι διαθέσιμες γλώσσες είναι αποθηκευμένες στον πίνακα

```
window['BlocklyGamesLanguages'] = ['el', 'en' ];
```

του boot.js, ενώ ορισμός της προεπιλογής γίνεται στο ίδιο αρχείο με τον κώδικα

```
lang = 'el';
window['BlocklyGamesLang'] = lang;
```

Στην γραμμή πλοήγησης υπάρχει το μενού επιλογής γλώσσας το οποίο εμφανίζει τις διαθέσιμες γλώσσες. Η αλλαγή γλώσσας γίνεται με την συνάρτηση BlocklyGames.changeLanguage() ο κώδικας της οποίας βρίσκεται στο αρχείο lib-games.js.

Κώδικας 16: Συνάρτηση αλλαγής γλώσσας με την χρήση του μενού επιλογής

```
BlocklyGames.changeLanguage = function() {
    var languageMenu = document.getElementById('languageMenu');
    var newLang = encodeURIComponent(
        languageMenu.options[languageMenu.selectedIndex].value);
    var search = window.location.search;
    if (search.length <= 1) {
        search = '?lang=' + newLang;
    } else if (/[?&]lang=[^&]*/.test(search)) {
        search = search.replace(/([?&]lang=)[^&]*/, '$1' + newLang);
    } else {
        search = search.replace(/\?/, '?lang=' + newLang + '&');
    }

    window.location = window.location.protocol + '//' +
        window.location.host + window.location.pathname + search;
};
```

Ενώ με την συνάρτηση BlocklyInterface.changeLanguage() αποθηκεύονται τα μπλοκ του χρήστη και το παιχνίδι φορτώνει με τη νέα γλώσσα. Ο κώδικας της συνάρτησης αυτής βρίσκεται στο αρχείο lib-interface.js. Στο σώμα της συνάρτησης αυτής καλούνται δυο συναρτήσεις, η πρώτη είναι η συνάρτηση BlocklyInterface.saveToSessionStorage() που αποθηκεύει τοπικά τα μπλοκ του χρήστη και η δεύτερη είναι η συνάρτηση για την αλλαγή της γλώσσας που περιγράψαμε παραπάνω.

Κώδικας 17: Επαναφόρτωση του παιχνιδιού με διαφορετική γλώσσα

```
BlocklyInterface.changeLanguage = function() {
    BlocklyInterface.saveToSessionStorage();
    BlocklyGames.changeLanguage();
};
```

Η κλήση της συνάρτησης BlocklyInterface.changeLanguage γίνεται στο ίδιο αρχείο, στο κύριο σώμα της συνάρτησης BlocklyInterface.init με τον κώδικα

Κώδικας 18: Κλήση της συνάρτησης BlocklyInterface.changeLanguage

```
var languageMenu = document.getElementById('languageMenu');
if (languageMenu) {
    languageMenu.addEventListener('change',
        BlocklyInterface.changeLanguage, true);
}
```

Με την μέθοδο document.getElementById(‘languageMenu’) επιλέγεται το στοιχείο μενού επιλογής γλώσσας της εφαρμογής. Σε αυτό προσθέτουμε ένα event handler με την μέθοδο addEventListener, κάθε φορά που θα ανιχνεύεται αλλαγή (το συμβάν είναι το ‘change’) στο

στοιχείο μενού επιλογής γλώσσας θα καλείται η συνάρτηση BlocklyInterface.changeLanguage.

Με την παραπάνω διαδικασία επιλέγεται η επιθυμητή γλώσσα από τον χρήστη και γίνεται η φόρτωση του παιχνιδιού στη νέα γλώσσα χωρίς να χαθεί η πορεία του χρήστη.

5.3.3. Κείμενα και μηνύματα του παιχνιδιού

Στο πρότυπο messages (template .messages), στο σώμα των προτύπων Closure βρίσκονται τα περισσότερα από τα κείμενα του παιχνιδιού που αφορούν τίτλους ή κείμενα στα μπλοκ, όπως για παράδειγμα το κείμενο «move forward» είτε τη βοήθεια εργαλείων που εμφανίζεται ενώ ο κέρσορας αιωρείται πάνω από ένα μπλοκ (tooltip). Για κάθε ένα από αυτά τα κείμενα δημιουργείται ένα μοναδικό αναγνωριστικό (id) και το κείμενο δίνεται μέσα τις ετικέτες {msg}….{/msg}. Για να οριστεί σωστά το κείμενο πρέπει να δοθούν δυο παράμετροι στην msg:

- παράμετρος meaning = ‘ ’ : περιέχει ένα όνομα στο οποίο θα αντιστοιχίζονται όλες οι μεταφράσεις του συγκεκριμένου κειμένου στα αρχεία json
- παράμετρος desc= ‘ ’: περιέχει μια μικρή περιγραφή του τι αφορά το κείμενο και είναι κυρίως συμβουλή προς τον μεταφραστή.

Για να γίνει κατανοητή η διαδικασία με την οποία εμφανίζεται ένα τέτοιο κείμενο στο παιχνίδι θα δώσουμε το παράδειγμα της βοήθειας εργαλείου του μπλοκ move forward. Το κείμενο που εμφανίζεται όταν αιωρείται ο κέρσορας πάνω από το μπλοκ ορίζεται πρώτα στο πρότυπο messages του Maze.soy:

```
<span id="Maze_moveForwardTooltip">{msg meaning="Maze.moveForwardTooltip" d
esc="tooltip - Moves the icon on the screen representing the player forward
 one square on the maze board."}Moves the player forward one space.{/msg}</
span>
```

Οι πληροφορίες που περιέχονται είναι:

- αναγνωριστικό μηνύματος: Maze_moveForwardTooltip
- όνομα μηνύματος: Maze.moveForwardTooltip
- βοήθεια για μεταφραστή: tooltip - Moves the icon on the screen representing the player forward one square on the maze board.
- Κείμενο μηνύματος που θα εμφανίζεται: Moves the player forward one space.

Στον φάκελο json υπάρχουν τα αρχεία en.json, el.json, keys.json, qqq.json . Στο πρώτο αρχείο βρίσκονται τα μηνύματα των προτύπων Closure στα αγγλικά, στο δεύτερο είναι η μετάφρασή τους στα Ελληνικά ενώ τα δυο τελευταία αρχεία δημιουργούνται με το make maze ή make turtle αντίστοιχα. Τα αρχεία json είναι ένα είδος λεξικού για την JavaScript,

αποτελούνται από ζεύγη κλειδί : τιμή χωρισμένα με άνω κάτω τελεία, ενώ μπορούμε να έχουμε όσα τέτοια ζεύγη θέλουμε στο ίδιο αρχείο χωρισμένα με κόμμα. Το μήνυμα πρέπει να οριστεί και στα δυο πρώτα αρχεία json, η αντιστοίχιση γίνεται χρησιμοποιώντας το όνομα του μηνύματος ως κλειδί όπως ορίστηκε στο πρότυπο messages και το ίδιο το μήνυμα ως τιμή του κλειδιού. Για το παράδειγμά μας έχουμε:

αρχείο en.json:

```
"Maze.moveForwardTooltip": "Moves the player forward one space."
```

αρχείο el.json:

```
"Maze.moveForwardTooltip": "Κινεί τον χαρακτήρα μία θέση μπροστά."
```

Τέλος το μήνυμα πρέπει να οριστεί και στο αρχείο js που ορίζονται τα μπλοκ, για το παράδειγμά μας είναι το block.js του φακέλου με διαδρομή appengine/maze/js. Η αντιστοίχιση του μηνύματος γίνεται μέσω της συνάρτησης BlocklyGames.getMsg('id') η οποία δέχεται ως παράμετρο το id του μηνύματος που θέλουμε να αντιστοιχίσουμε στο συγκεκριμένο tooltip.

Κώδικας 19: Tooltip του μπλοκ move forward

```
50  Blockly.Blocks['maze_moveForward'] = {
51      /**
52       * Block for moving forward.
53       * @this {Blockly.Block}
54       */
55      init: function() {
56          this.jsonInit({
57              "message0": BlocklyGames.getMsg('Maze_moveForward'),
58              "previousStatement": null,
59              "nextStatement": null,
60              "colour": Maze.Blocks.MOVEMENT_HUE,
61              "tooltip": BlocklyGames.getMsg('Maze_moveForwardTooltip')
62          });
63      }
64  };
```

Με αυτό τον τρόπο στο μπλοκ maze_moveForward αντιστοιχούμε (Κώδικας 19, γραμμή 61) την βοήθεια εργαλείου με αναγνωριστικό Maze_moveForwardTooltip. Το αποτέλεσμα φαίνεται στην παρακάτω εικόνα:

Εικόνα 44: Βοήθεια εργαλείου για το μπλοκ move forward στα αγγλικά (πάνω εικόνα) και στα ελληνικά (κάτω εικόνα)

Την ίδια λογική ακολουθούν όλα τα μηνύματα και οι οδηγίες του παιχνιδιού, δηλαδή κάθε μήνυμα έχει το δικό του αναγνωριστικό, ορίζεται μέσα στις ετικέτες {msg} με τις παραμέτρους meaning, desc, καταχωρείται στα αρχεία en.json και el.json και εμφανίζεται στο παιχνίδι μέσω κατάλληλα τοποθετημένων συναρτήσεων στα αρχεία js.

5.3.4. Επιλογή εμφάνισης επιπέδου στον Λαβύρινθο

Όπως αναφέρθηκε και στο Κεφάλαιο 4, στα επίπεδα 1-6 η εμφάνιση του επιπέδου (skin) αποτελείται από το φόντο (background) που είναι μια εικόνα 400 x 400 εικονοστοιχείων , το μονοπάτι του λαβυρίνθου (tiles) που δείχνει όλες τις πιθανές διασταυρώσεις του λαβυρίνθου και τον χαρακτήρα του παίχτη αποτελείται από ένα αρχείο png αποτελούμενο από 21 ισομεγέθη πλαίσια με το χαρακτήρα στη μέση (sprite). Όπως φαίνεται και στην Εικόνα 45 ο χαρακτήρας ξεκινά με γυρισμένη την πλάτη, στα επόμενα πλαίσια βλέπουμε την στροφή του προς τα δεξιά με το 5^{ο} πλαίσιο να έχει το δεξί προφίλ, το 9^{ο} πλαίσιο να έχει το μπροστινό μέρος του και το 13^{ο} το αριστερό προφίλ. Το τελευταίο πλαίσιο είναι και το εικονίδιο επιλογής του χαρακτήρα στο πτυσσόμενο μενού στα αριστερά της μπάρας πλοήγησης.

Εικόνα 45: Εικονίδια χαρακτήρων παιχνιδιού

Οι χαρακτήρες είναι αστροναύτης, ζόμπι, μέλισσα. Τα γραφικά των δυο τελευταίων, το φόντο και το μονοπάτι είναι ανοιχτού κώδικα και τα βρήκαμε στο GitHub https://github.com/CelineDknp/JobBlockly .

Ο χαρακτήρας του αστροναύτη και το μονοπάτι υπήρχαν ήδη στα παιχνίδια Blockly και τροποποιήθηκαν με την βοήθεια του λογισμικού Gimp. Ενώ το φόντο της πρώτης εμφάνισης επιπέδου πάρθηκε από την ιστοσελίδα https://pixabay.com/photos/orion-nebula-emission-nebula-11107/ και το εικονίδιο στο τέλος του μονοπατιού από την ιστοσελίδα https://www.freepnglogos.com/images/rocket-19743.html .

Τα γραφικά βρίσκονται στον πίνακα Maze.SKINS και είναι τύπου αντικειμένου (object) με ιδιότητες όπως sprite (η εικόνα με 21 πλαίσια), avatar (εικόνα που εμφανίζεται στο μενού επιλογών χαρακτήρα), tiles (το μονοπάτι του λαβυρίνθου), marker (το αντικείμενο στο τέλος του μονοπατιού) . Τα αρχεία με κατάληξη .ogg και .mp3 είναι για τους ήχους που αναπαράγονται όταν ο χαρακτήρας βρίσκει εμπόδιο, χάνει ή κερδίζει το επίπεδο.

Κώδικας 20: αντικείμενο του πίνακα Maze.SKINS

```
{
    sprite: 'maze/bee2.png',
    avatar: 'maze/static_bee2.png',
    tiles: 'maze/tiles_bee.png',
    marker: 'maze/marker_honey.png',
    background: 'maze/bg_bee.png',
    look: '#000',
    winSound: ['maze/win.mp3', 'maze/win.ogg'],
    crashSound: ['maze/fail_panda.mp3', 'maze/fail_panda.ogg'],
    crashType: Maze.CRASH_FALL
}
```

Στον λαβύρινθο η αλλαγή χαρακτήρα μπορεί να γίνει κατά την διάρκεια του παιχνιδιού χωρίς να χαθεί η πρόοδος του παιχνιδιού, η συνάρτηση Maze.changePegman είναι υπεύθυνη για την αλλαγή αυτή.

Κώδικας 21: Συνάρτηση Maze.changePegman

```
 * Reload with a different Pegman skin.
 * @param {number} newSkin ID of new skin.
 */
Maze.changePegman = function(newSkin) {
    BlocklyInterface.saveToSessionStorage();
    location = location.protocol + '//' + location.host + location.pathname +
        '?lang=' + BlocklyGames.LANG + '&level=' + BlocklyGames.LEVEL +
        '&skin=' + newSkin;
};
```

Η δημιουργία λαβυρίνθου κάθε επιπέδου γίνεται στον πίνακα Maze.map . Τα στοιχεία αυτού του πίνακα είναι τύπου Πίνακας (Array) με 8 γραμμές και 8 στήλες ο κάθε ένας που αντιπροσωπεύουν το κυρίως πλαίσιο κάθε επιπέδου.

Κώδικας 22: απόσπασμα του πίνακα Maze.map

```
Maze.map = [
    // Level 0.
    undefined,
    // Level 1.
    [
        [0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 1, 3, 0, 0],
        [0, 0, 0, 0, 1, 0, 0, 0],
        [0, 0, 0, 2, 1, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0]
    ],
```

Κάθε ένας από αυτούς τους φωλιασμένους πίνακες στον Maze.map παριστάνει ένα επίπεδο και σε κάθε γραμμή του φωλιασμένου πίνακα τα στοιχεία μπορούν να πάρουν τιμές 0,1,2,3,5 :

- 0: τοίχος
- 1: ανοιχτή διαδρομή
- 2: εκκίνηση
- 3: τερματισμός
- 5: βρόχος

Τα παραπάνω στον κώδικα αποτελούν ιδιότητες του αντικειμένου Maze.SquareType.

Η συνάρτηση Maze.drawMap είναι υπεύθυνη για την δημιουργία των γραφικών του λαβυρίνθου σύμφωνα με τις οδηγίες που έχουν δοθεί στον πίνακα Maze.map . Ο εντοπισμός του τέλους και της αρχής γίνεται σε έναν βρόχο επανάληψης εκτός της συνάρτησης.

Κώδικας 23: Εντοπισμός αρχής, τέλους και κλειστής διαδρομής

```
for (var y = 0; y < Maze.ROWS; y++) {
    for (var x = 0; x < Maze.COLS; x++) {
        if (Maze.map[y][x] == Maze.SquareType.START) {
            Maze.start_ = { x: x, y: y };
            //το επόμενο if else δημιουργεί κλειστή διαδρομή
        } else if (Maze.map[y][x] == Maze.SquareType.LOOP) {
            Maze.start_ = { x: x, y: y };
            Maze.finish_ = { x: x, y: y };
        } else if (Maze.map[y][x] == Maze.SquareType.FINISH) {
            Maze.finish_ = { x: x, y: y };
        } else if (Maze.map[y][x] == Maze.SquareType.OBSTACLE) {
            Maze.result = Maze.ResultType.FAILURE;
        }
    }
}
```

5.3.5. Επίπεδα Ζωγραφικής

Στα επίπεδα της Ζωγραφικής δεν υπάρχει επιλογή εμφάνισης, ο χρήστης πρέπει να χρησιμοποιήσει τα διαθέσιμα μπλοκ για να σχεδιάσει το σχήμα που φαίνεται στο πλαίσιο του παιχνιδιού. Για να δημιουργήσουμε τα κατάλληλα σχήματα χρησιμοποιήσαμε τις συναρτήσεις Turtle.answer() στο answer.js και Turtle.drawAnswer() στο turtle.js . Στον Κώδικας 24 παρουσιάζουμε τον κώδικα για τις λύσεις των επιπέδων στο aMazeD. Χρησιμοποιούμε την switch με παράμετρο το τρέχον επίπεδο για να δούμε σε ποιο case ανήκει. Κάθε case αντιπροσωπεύει ένα από τα επίπεδα της Ζωγραφικής και περιέχει τον κώδικα για να σχεδιαστεί το επιθυμητό σχήμα κάνοντας χρήση βρόγχων, συναρτήσεων όπως οι move και turn και βάζοντας στην περίπτωση των επιπέδων 7 και 10 και χρώμα στο σχέδιο. Ο χρήστης μπορεί να χρησιμοποιήσει το μπλοκ «Κάνε το χρώμα» για να αλλάξει το χρώμα του πινέλου του, ωστόσο η χρωματική επιλογή δεν ελέγχεται στην ορθότητα της λύσης. Έπειτα ο κώδικας της Turtle.answer χρησιμοποιείται στην Turtle.drawAnswer για να δημιουργηθεί το κατάλληλο σχήμα στο επίπεδο και η τελευταία καλείται στην Turtle.init για να φορτώνει κάθε φορά το κατάλληλο σχήμα στο επίπεδο.

Κώδικας 24: Η συνάρτηση Turtle.answer

```
Turtle.answer = function() {
    switch (BlocklyGames.LEVEL) {
        case 7:
            // Square.
            Turtle.penColour('#ff0000');
            for (var count = 0; count < 4; count++) {
                Turtle.move(100);
                Turtle.turn(90);
            }
            break;
        case 8:
            // ορθογώνιο.
            for (var count = 0; count < 2; count++) {
                Turtle.move(50);
                Turtle.turn(90);
                Turtle.move(100);
                Turtle.turn(90);
            }
            break;
        case 9:
            // ισόπλευρο τρίγωνο.
            for (var count = 0; count < 3; count++) {
                Turtle.move(50);
                Turtle.turn(120);
            }
            break;
        case 10:
            Turtle.penColour('#ffff00');
            for (var count = 0; count < 4; count++) {
                for (var j = 0; j < 3; j++) {
                    Turtle.move(50);
                    Turtle.turn(120);
                }
                Turtle.turn(90);
            }
            break;
    }
};
```

5.3.6. Εισαγωγή της βιβλιοθήκης Blockly, εργαλειοθήκη και χώρος εργασίας στη διεπαφή του aMazeD

Στην διεπαφή του χρήστη βρίσκεται το κυρίως πλαίσιο του παιχνιδιού που δημιουργείται με την χρήση της ετικέτας div και το αναγνωριστικό visualization στα αρχεία Maze.soy και Turtle.soy. Στα ίδια αρχεία δημιουργείται και το div με το αναγνωριστικό blockly στο οποίο θα γίνει η εισαγωγή της βιβλιοθήκης Blockly (injection). Η διαδικασία υλοποιείται στις συναρτήσεις Maze.init και Turtle.init, παραθέτουμε τον κώδικα στην Maze.init στο στιγμιότυπο Κώδικας 25.

Κώδικας 25: Injection of Blockly στη συνάρτηση Maze.init

```
570   var blocklyDiv = document.getElementById('blockly');
571   var visualization = document.getElementById('visualization');
572   var onresize = function(e) {
573       var top = visualization.offsetTop;
574       blocklyDiv.style.top = Math.max(10, top - window.pageYOffset) +
      'px';
575       blocklyDiv.style.left = rtl ? '10px' : '420px';
576       blocklyDiv.style.width = (window.innerWidth - 440) + 'px';
577   };
578   window.addEventListener('scroll', function() {
579       onresize(null);
580       Blockly.svgResize(BlocklyInterface.workspace);
581   });
582   window.addEventListener('resize', onresize);
583   onresize(null);
584
585
586   BlocklyInterface.injectBlockly({
587       'maxBlocks': Maze.MAX_BLOCKS,
588       'rtl': rtl,
589       'trashcan': true,
590       'zoom': true
591   });
592   BlocklyInterface.workspace.getAudioManager().load(Maze.SKIN.winSound,
      'win');
593   BlocklyInterface.workspace.getAudioManager().load(Maze.SKIN.
      crashSound, 'fail');
594   // Not really needed, there are no user-defined functions or
      variables.
595   Blockly.JavaScript.addReservedWords('moveForward,moveBackward,' +
596       'turnRight,turnLeft,isPathForward,isPathRight,isPathBackward,
      isPathLeft');
```

Αρχικά ορίζονται οι μεταβλητές που επιλέγουν με την μέθοδο getElementById τα div με τα κατάλληλα αναγνωριστικά. Έπειτα ορίζονται τα άνω περιθώρια και το πλάτος του κυρίως πλαισίου. Τέλος γίνεται η εισαγωγή της βιβλιοθήκης στο στοιχείο ‘blockly’ μέσω της συνάρτησης BlocklyInterface.injectBlockly .

Εργαλειοθήκη (toolbox)

Τα μπλοκ του παιχνιδιού που χρησιμοποιούνται για την κίνηση του χαρακτήρα βρίσκονται στην εργαλειοθήκη της διεπαφής (toolbox). Ο παίκτης έχει τη δυνατότητα να χρησιμοποιήσει οποιοδήποτε μπλοκ στην εργαλειοθήκη του για να συμπληρώσει τον υπάρχον κώδικα ή να τον σχηματίσει από το μηδέν. Ο ορισμός αυτών των μπλοκ γίνεται στα Maze.soy και Turtle.soy στο πρότυπο toolbox.

Κώδικας 26: Εργαλειοθήκη για τα επίπεδα του Λαβυρίνθου

```
{template .toolbox}
  <xml id="toolbox" xmlns="https://developers.google.com/blockly/xml">

    <block type="maze_moveForward"></block>
    <block type="maze_turn"><field name="DIR">turnLeft</field></block>
    <block type="maze_turn"><field name="DIR">turnRight</field></block>
    {if $ij.level == 3 or $ij.level == 7}
    //add repeat n times
      <block type="controls_repeat_ext">
        <value name="TIMES">
          <shadow type="math_number">
            <field name="NUM">3</field>
          </shadow>
        </value>
      </block>
    {/if}
    {if $ij.level >= 4 or $ij.level == 2}
      <block type="maze_if"></block>
      <block type="maze_forever"></block>
       <block type="maze_if"><field name="DIR">isPathLeft</field></block>
      {if $ij.level > 6}
        <block type="maze_ifElse"></block>
      {/if}
    {/if}
  </xml>
{/template}
```

Ο ορισμός των μπλοκ και της λειτουργικότητας που εκτελούν γίνεται στα αρχεία block.js , για τον Λαβύρινθο και τη Ζωγραφική τα μπλοκ είναι διαφορετικά ωστόσο υπάρχουν κοινά μπλοκ όπως το επανέλαβε ν-φορές. Ο ορισμός και η λειτουργικότητα του μπλοκ παρουσιάζονται στα στιγμιότυπα Κώδικας 27 και Κώδικας 28.

Κώδικας 27: Ορισμός μπλοκ Επανέλαβε ν-φορές

```
Blockly.Blocks['controls_repeat_ext'] = {
    /**
     * Block for repeat n times (internal number).
     * @this {Blockly.Block}
     */
    init: function() {
        this.jsonInit({
            "message0": Blockly.Msg['CONTROLS_REPEAT_TITLE'],
            "args0": [{
                "type": "field_dropdown",
                "name": "TIMES",
                "options": [
                    ["3", "3"],
                    ["4", "4"],
                    ["5", "5"]
                ]
            }],
            "previousStatement": true,
            "nextStatement": null,
            "colour": 60,
            "tooltip": Blockly.Msg['CONTROLS_REPEAT_TOOLTIP'],
            "helpUrl": Blockly.Msg['CONTROLS_REPEAT_HELPURL']
        });
        this.appendStatementInput('DO')
            .appendField(Blockly.Msg['CONTROLS_REPEAT_INPUT_DO'])
            ;
    }
};
```

Ο ορισμός ενός μπλοκ γίνεται με την μέθοδο jsonInit της κλάσης Block, η οποία δέχεται ως όρισμα ένα αντικείμενο τύπου json με ζεύγη κλειδί/τιμή

Πίνακας 10: Ιδιότητες αντικειμένου json για τον ορισμό των μπλοκ

Όνομα κλειδιού	Περιγραφή
message0	Το όνομα που θα εμφανίζεται στο μπλοκ
args0	η συνθήκη που εμφανίζεται στο μπλοκ, στον Κώδικας 27 είναι πτυσσόμενη λίστα
previousStatement	δυνατότητα του μπλοκ να συνδέεται με άλλα μπλοκ στο πάνω μέρος
nextStatement	δυνατότητα του μπλοκ να συνδέεται με άλλα μπλοκ στο κάτω μέρος
color	το χρώμα του μπλοκ
tooltip	η βοήθεια που εμφανίζεται όταν αιωρείται το ποντίκι πάνω από το μπλοκ

Κώδικας 28: Λειτουργικότητα μπλοκ Επανέλαβε ν-φορές

```
Blockly.JavaScript['controls_repeat_ext'] = function(block) {
    // Repeat n times.
    if (block.getField('TIMES')) {
        // Internal number.
        var repeats = String(Number(block.getFieldValue('TIMES'))
        );
    } else {
        // External number.
        var repeats = Blockly.JavaScript.valueToCode(block,
        'TIMES',
            Blockly.JavaScript.ORDER_ASSIGNMENT) || '0';
    }
    var branch = Blockly.JavaScript.statementToCode(block, 'DO');
    branch = Blockly.JavaScript.addLoopTrap(branch, block);
    var code = '';
    var loopVar = Blockly.JavaScript.variableDB_.getDistinctName(
        'count', Blockly.VARIABLE_CATEGORY_NAME);
    var endVar = repeats;
    if (!repeats.match(/^\w+$/) && !Blockly.isNumber(repeats)) {
        endVar = Blockly.JavaScript.variableDB_.getDistinctName(
            'repeat_end', Blockly.VARIABLE_CATEGORY_NAME);
        code += 'var ' + endVar + ' = ' + repeats + ';\n';
    }
    code += 'for (var ' + loopVar + ' = 0; ' +
        loopVar + ' < ' + endVar + '; ' +
        loopVar + '++) {\n' +
        branch + '}\n';
    return code;
};
```

Χώρος εργασίας (workspace)

Στην έκδοση 1 του παιχνιδιού ο χώρος εργασίας έχει κάποια μπλοκ όπως παρουσιάστηκε και στο κεφάλαιο 4. Τα μπλοκ στον χώρο εργασίας εμφανίζονται με την χρήση την μέθοδο loadBlocks του στιγμιότυπου BlocklyInterface, η μέθοδος ορίζεται μαζικά για όλα τα παιχνίδια στο αρχείο js/lib-interface.js. Η συνθήκη επιλογής επιπέδου και η χρήση της μεθόδου δίνονται από τον κώδικα

```
if (BlocklyGames.LEVEL == 6) {
        BlocklyInterface.loadBlocks(defaultXml6, false);
    }
```

Το πρώτο όρισμα της μεθόδου είναι μια τοπική μεταβλητή τύπου string που περιέχει τα μπλοκ που θέλουμε να βάλουμε στον χώρο εργασίας, το δεύτερο όρισμα είναι μια μεταβλητή τύπου Boolean όταν παίρνει τιμή true εμφανίζει τα μπλοκ του προηγούμενου επιπέδου, θα μπορούσε να πάρει και σαν όρισμα μια συνάρτηση που να τροποποιεί τα μπλοκ που κληρονομούνται.

Κώδικας 29: ορισμός μεταβλητής defaultXml6

```
736     var defaultXml6 =
737         '<xml>' +
738         '<block type="maze_forever" x="70" y="70">' +
739         '<statement name="DO">' +
740         ' <block type="maze_ifElse">' +
741         '<statement name="DO">' +
742         ' <block type="maze_moveForward">' +
743         '</block>' + //end 1st block
744         '</statement>' + //end do-statement
745         '<statement name="ELSE">' +
746         ' <block type="maze_ifElse"><field name="DIR">isPathRight</field>' +
747         '<statement name="DO">' +
748         '<block type="maze_turn"><field name="DIR">turnRight</field> ' +
749         '</block>' + //end block
750         '</statement>' + //end do-statement
751         '<statement name="ELSE">' +
752         '</statement>' +
753         '</block>' +
754         '</statement>' + //end else-statement
755         '</block>' + //end external ifElse_path
756         '</statement>' +
757         '</block>' + //end maze_forever
758         '</xml>';
```

Στο στιγμιότυπο Κώδικας 29 παρουσιάζεται η μεταβλητή defaultXml6 για το επίπεδο 6 του Λαβυρίνθου όπως ορίζεται στο maze.js, τα μπλοκ που εμφανίζονται στον χώρο εργασίας μπορούμε να τα δούμε στην Εικόνα 31 του κεφαλαίου 4. Η μεταβλητή defaultXml για κάθε επίπεδο είναι τύπου αλφαριθμητικό και παράγει ένα κείμενο xml . Το κείμενο xml χρησιμοποιείται για την κατασκευή δεδομένων για αποθήκευση και μεταφορά. Τα δεδομένα που δημιουργούνται εδώ είναι τα μπλοκ τα οποία θα εμφανίζονται στον χώρο εργασίας. Στο συγκεκριμένο παράδειγμα (Κώδικας 29) υπάρχουν πολλά φωλιασμένα μπλοκ: ξεκινάμε με το μπλοκ Επανέλαβε μέχρι…τέρμα στην γραμμή 738, το οποίο θα επαναλαμβάνει τον κώδικα ανάμεσα στις ετικέτες <statement name=”Do”>..</statement> των γραμμών 739 και 756 μέχρι ο χαρακτήρας να φτάσει στο τέρμα. Γενικότερα στα μπλοκ που έχουν συνθήκη, όπως στο παράδειγμά μας, μέσα στις ετικέτες <statement name=”Do”> βρίσκονται οι εντολές του κάνε ενώ με ετικέτα <statement name=”Else”> ξεκινούν οι εντολές του αλλιώς.

Για να στοιβάξουμε 2 ή περισσότερα μπλοκ πρέπει να χρησιμοποιήσουμε την ετικέτα <next>…</next> ανάμεσα στη δήλωση μπλοκ. Για παράδειγμα στο επίπεδο 9 έχουμε τον Κώδικας 30 που όπως φαίνεται και στην Εικόνα 35 του Κεφαλαίου 4 έχει ένα μπλοκ επανέλαβε ν-φορές και στο κάνε εμφανίζονται δυο μπλοκ τα κινήσου μπροστά και στρίψε δεξιά. Τα μπλοκ στοιβάζονται με την χρήση του next. Αρχικά στην γραμμή 287 ορίζεται το πρώτο μπλοκ με την ετικέτα <block> και την ιδιότητα type για να προσδιοριστεί το είδος

του μπλοκ, αφού οριστεί και η τιμή της επιλογής με την ετικέτα <field> και χωρίς να κλείσει η ετικέτα <block> μέσα στις <next>…</next> των γραμμών 289-293 ορίζεται το 2ο στην στοίβα μπλοκ που είναι το στρίψε δεξιά. Έπειτα κλείνομε με </block> το 1ο μπλοκ στη στοίβα και τέλος κλείνουμε το statement στην γραμμή 295. Τέλος κλείνουμε την ετικέτα του μπλοκ επανέλαβε στην γραμμή 296.

Κώδικας 30: Κώδικας για εμφάνιση των μπλοκ του επιπέδου 9

```
282     var defaultXml9 =
283         '<xml>' +
284         '<block type="turtle_repeat_internal" x="70" y="70">' +
285         '<field name="TIMES">3</field>' +
286         '<statement name="DO">' +
287         '<block type="turtle_move_internal">' +
288         '<field name="VALUE">50</field>' +
289         '<next>' +
290         '<block type="turtle_turn_internal">' +
291         '<field name="VALUE"> </field>' +
292         '</block>' + // τέλος 2ου block
293         '</next>' +
294         '</block>' + // τέλος πρώτου block
295         '</statement>' +
296         '</block>' + // τέλος repeat
297         '</xml>';
```

5.4. Χρονόμετρο

Ο χρόνος κάθε επιπέδου εμφανίζεται στην πάνω αριστερή γωνία στο κυρίως πλαίσιο. Για το πλαίσιο του χρονομέτρου χρησιμοποιήθηκε το στοιχείο div του Maze.soy με αναγνωριστικό capacityBubble που εξυπηρετεί διαφορετικό σκοπό στα Παιχνίδια Blockly και δημιουργήθηκε ένα ίδιο div και στο Turtle.soy . Ο κώδικας στα πρότυπα Closure είναι:

```
<div id="capacityBubble">
      <div id="capacity">Write something</div>
    </div>
</div>
```

Έπειτα δημιουργήθηκε η συνάρτηση Maze.setTimerLevel() και αντίστοιχα η Turtle.setTimerLevel που έχουν τον ίδιο κώδικα, όπως παρουσιάζεται στο στιγμιότυπο Κώδικας 31 .

Κώδικας 31: Συνάρτηση για την δημιουργία χρονομέτρου

```
467  Maze.setTimerLevel = function() {
468      var tick = function() {
469          var min = String(Math.trunc(time / 60)).padStart(2, 0);
470          var sec = String(time % 60).padStart(2, 0);
471          // In each call, print the remaining time to UI
472          document.getElementById('capacity').textContent = min + ":" + sec;
473          time++;
474      };
475      var time = 0;
476      // Call the timer every second
477      tick();
478      timer = setInterval(tick, 1000);
479
480      return timer;
481  }
482
483  var timer;
```

Στη συνάρτηση αυτή ορίζεται η μεταβλητή time που έχει αρχική τιμή 0. Μέσα στη συνάρτηση tick που καλείται για πρώτη φορά μετά τον ορισμό της time η μεταβλητή αυξάνεται κατά 1 και ορίζονται οι μεταβλητές min, ως το ακέραιο μέρος της διαίρεσης της time με το 60, και sec, ως το υπόλοιπο της διαίρεσης της time με το 60. Και οι δυο μεταβλητές είναι αλφαριθμητικού τύπου και τροποποιούνται με την μέθοδο padStart στο να έχουν πάντα δυο ψηφία και στην περίπτωση που υπάρχει ένα ψηφίο προστίθεται στην αρχή του αλφαριθμητικού ένα 0. Έπειτα η tick επιλέγει το div που δημιουργήσαμε για το χρονόμετρο και εμφανίζει την τιμή των min, sec. Τέλος η tick καλείται κάθε ένα δευτερόλεπτο με την μέθοδο setInterval, άρα κάθε δευτερόλεπτο η time αυξάνει κατά ένα. Στην πράξη η συνάρτηση setTimerLevel επιστρέφει την μέθοδο setInterval(tick,1000).
Η κλήση της συνάρτησης Maze.setTimerLevel γίνεται στο σώμα της Maze.init (ή Turtle.init) αντίστοιχα με τον κώδικα

```
if (timer) clearInterval(timer);
    timer = Maze.setTimerLevel();
```

Στον κώδικα αυτό ελέγχουμε αν υπάρχει ήδη κάποιο timer που τρέχει και αν ναι το μηδενίζουμε με την clearInterval μέθοδο και αποθηκεύουμε στην μεταβλητή timer την μέθοδο που επιστρέφει η setTimerLevel. Έτσι κάθε φορά που γίνεται ανανέωση της σελίδας ή που ο παίκτης επιστρέφει σε μη ολοκληρωμένο επίπεδο το χρονόμετρο ξεκινά από το μηδέν.

5.5. Πλήκτρα Παιχνιδιού και γεννήτρια κώδικα

5.5.1. Συναρτήσεις για την κίνηση του χαρακτήρα σε Λαβύρινθο

Η κίνηση του χαρακτήρα στο aMazeD βασίζεται στην μετάφραση του κώδικα που δημιουργεί ο χρήστης με τα μπλοκ, στην γλώσσα προγραμματισμού JavaScript και η υλοποίηση της κίνησης που περιγράφεται στον κώδικα αυτό. Στο Κεφάλαιο 2 αναφέραμε ότι η βιβλιοθήκη της Blockly παρέχει τη δυνατότητα εξαγωγής του κώδικα του συντάκτη της Blockly σε διάφορες γλώσσες προγραμματισμού και στο δικιά μας περίπτωση στην γλώσσα JavaScript, αυτό επιτυγχάνεται με την χρήση του διερμηνέα της JavaScript, JS-Interpreter, που αναπτύχθηκε από τον Neil Fraser.

Στα επίπεδα του Λαβυρίνθου οι συναρτήσεις που είναι υπεύθυνες για την ανάγνωση του κώδικα του χρήστη και την προσωρινή αποθήκευσή του είναι οι Maze.play και Maze.execute. Οι συναρτήσεις αυτές έχουν ένα κοινό κομμάτι κώδικα που παρουσιάζεται στο στιγμιότυπο Κώδικας 32. Στην περίπτωση της Maze.play δεν έχουμε εκχώρηση του αποτελέσματος ΕΠΙΤΥΧΙΑ/ΑΠΟΤΥΧΙΑ (SUCCESS / FAILURE) ανά επίπεδο, αυτή είναι και η ειδοποιός διαφορά των δυο συναρτήσεων, οι οποίες κάνουν ακριβώς την ίδια διαδικασία μόνο που η execute εκχωρεί το αποτέλεσμα στην μεταβλητή Maze.result, ενώ η play απλά καλεί την εκτέλεση του κώδικα χωρίς να αποθηκεύει το αποτέλεσμα.

Στο στιγμιότυπο Κώδικας 32, παρατηρούμε ότι γίνεται η αρχικοποίηση του πίνακα των αποτελεσμάτων Maze.log στην γραμμή 1360. Επιλέγεται ο κώδικας του χρήστη μέσω της συνάρτησης BlocklyInterface.getJSCode() και δημιουργείται ένα νέο στιγμιότυπο διερμηνέα JS-Interpreter και αρχικοποιείται ο κατασκευαστής του με τον κώδικα του χρήστη (code) και την συνάρτηση Maze.initInterpreter(). Ο JS-Interpreter μπορεί να θεωρηθεί ένα κουτί (sandboxed) που είναι πλήρως απομονωμένο από τον φυλλομετρητή, κάθε μπλοκ που εκτελεί ενέργειες χρειάζεται την προσθήκη μιας API στον διερμηνέα, τον ρόλο αυτό στον Λαβύρινθο αναλαμβάνει η Maze.initInterpreter.

Στις γραμμές 1368-1389 μέσα σε ένα try…catch εκχωρείται στην Maze.result η κατάλληλη τιμή και επιστρέφεται μια Λογική τιμή Αληθής/Ψευδής για κανονικό τερματισμό και ένα Σφάλμα για μη κανονικό τερματισμό. Υπάρχουν 4 πιθανά αποτελέσματα:

1. Αν ο χαρακτήρας έχει φτάσει στην γραμμή τερματισμού [Επιτυχία] τότε η Λογική συνάρτηση Maze.notDone επιστρέφει Ψευδές και επομένως εκχωρείται η τιμή Maze.ResultType.SUCCESS (εναλλακτική υπόθεση) στην μεταβλητή Maze.result.

2. Αν το πρόγραμμα του χρήστη δεν κατάφερε να επιλύσει το Λαβύρινθο και τερματίστηκε χωρίς Σφάλματα τότε το Maze.Result παίρνει την τιμή Maze.ResultType.FAILURE .

Κώδικας 32: Κοινό κομμάτι κώδικα των συναρτήσεων Maze.execute και Maze.play[15]

```
1354    if (!('Interpreter' in window)) {
1355      // Interpreter lazy loads and hasn't arrived yet.  Try again later.
1356      setTimeout(Maze.execute, 250);
1357      return;
1358    }
1359
1360    Maze.log = [];
1361    Blockly.selected && Blockly.selected.unselect();
1362    var code = BlocklyInterface.getJsCode();
1363    BlocklyInterface.executedJsCode = code;
1364    BlocklyInterface.executedCode = BlocklyInterface.getCode();
1365    Maze.result = Maze.ResultType.UNSET;
1366    var interpreter = new Interpreter(code, Maze.initInterpreter);
1367
1368    try {
1369      var ticks = 10000; // 10k ticks runs Pegman for about 8 minutes.
1370      while (interpreter.step()) {
1371        if (ticks-- == 0) {
1372          throw Infinity;
1373        }
1374      }
1375      Maze.result = Maze.notDone() ?
1376          Maze.ResultType.FAILURE : Maze.ResultType.SUCCESS;
1377    } catch (e) {
1378      // A boolean is thrown for normal termination.
1379      // Abnormal termination is a user error.
1380      if (e === Infinity) {
1381        Maze.result = Maze.ResultType.TIMEOUT;
1382      } else if (e === false) {
1383        Maze.result = Maze.ResultType.ERROR;
1384      } else {
1385        // Syntax error, can't happen.
1386        Maze.result = Maze.ResultType.ERROR;
1387        alert(e);
1388      }
1389    }
```

3. Αν το πρόγραμμα έχει τερματιστεί επειδή έτρεχε για πολύ μεγάλο χρονικό διάστημα προκαλείται εξαίρεση και εκχωρείται η τιμή Maze.ResultType.TIMEOUT στο Maze.result.
4. Αν προκύψει κάποιο άλλο Σφάλμα προκαλείται εξαίρεση και εμφανίζεται το σφάλμα σε ένα πλαίσιο alert.

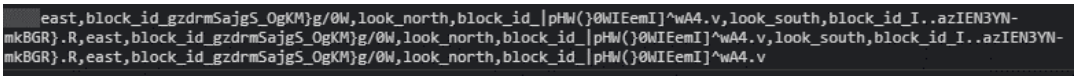
[S]Στην Maze.play η συνάρτηση επανάκλησης στην setTimeout στην γραμμή 1356 είναι η Maze.play και όχι η Maze.execute και δεν υπάρχει η έκφραση στις γραμμές 1375-1376.

Μετά την εκτέλεση του κώδικα στο try..catch, ο πίνακας Maze.log θα περιέχει όλες τις κινήσεις του χαρακτήρα (Εικόνα 46).

Κώδικας 33: Εκτέλεση κίνησης στην Maze.execute

```
1399    if (Maze.result == Maze.ResultType.SUCCESS) {
1400        Maze.stepSpeed = 50;
1401        Maze.log.push(['finish', null]);
1402        //αποθήκευσε δεδομένα επιτυχίας
1403        document.getElementById("msgSuccess").style.display = 'inline';
1404        Maze.levelData[BlocklyGames.LEVEL - 1].score = 10 * BlocklyGames.LEVEL;
1405        Maze.levelData[BlocklyGames.LEVEL - 1].result = 'Success';
1406    } else {
1407        Maze.stepSpeed = 100;
1408        //στην περίπτωση που δεν έχουμε success δεν θέλουμε να τρέχει τον κώδικα
1409        Maze.log.push(['end', null]);
1410        //αποθήκευσε δεδομένα αποτυχίας
1411        document.getElementById("msgFail").style.display = 'inline';
1412        Maze.levelData[BlocklyGames.LEVEL - 1].score = 0;
1413        Maze.levelData[BlocklyGames.LEVEL - 1].result = 'Failure';
1414    }
1415
1416
1417    // Maze.log now contains a transcript of all the user's actions.
1418    // Reset the maze and animate the transcript.
1419    Maze.reset(false);
1420    Maze.pidList.push(setTimeout(Maze.animate, 100));
```

Εικόνα 46: Εκτύπωση του Maze.log στην κονσόλα



Ο κώδικας στην Maze.result συνεχίζει με τον έλεγχο αν το αποτέλεσμα είναι Επιτυχία στην γραμμή 1399, αν αυτό είναι αληθές τότε αποθηκεύεται το αποτέλεσμα finish στον πίνακα Maze.log, η ταχύτητα εκτέλεσης γίνεται 50 (όσο πιο μικρή η τιμή της τόσο πιο γρήγορη είναι η εκτέλεση του κώδικα) και αποθηκεύονται τα κατάλληλα δεδομένα του επιπέδου. Αν ο έλεγχος στην 1399 δώσει Ψευδές τότε η εκτέλεση γίνεται με ταχύτητα 100 είναι δηλαδή πιο αργή για να μπορεί ο χρήστης να εντοπίσει τυχόν λάθη, στον πίνακα Maze.log αποθηκεύεται το αποτέλεσμα end και ενημερώνονται κατάλληλα τα δεδομένα του επιπέδου. Μετά από την εκτέλεση του if το Maze.log περιέχει όλες τις απαραίτητες ενέργειες και καλείται η Maze.animate (γραμμή 1420), ενώ στην γραμμή 1419 απενεργοποιείται η δυνατότητα επανεκκίνησης. Στην play δεν υπάρχει ο έλεγχος για το αποτέλεσμα του επιπέδου και η ταχύτητα εκτέλεσης είναι πάντα 150, αρκετά αργή για να προσφέρει την δυνατότητα στον χρήστη να δει την κίνηση σε σχέση με τον κώδικά του (debugging). Επιπλέον ο κώδικας στις γραμμές 1419-1420 παραμένει ο ίδιος δηλαδή καλείται η Maze.animate και απενεργοποιείται η δυνατότητα επανεκκίνησης.

Στα επίπεδα της Ζωγραφικής η συνάρτηση που είναι υπεύθυνη για την εκτέλεση του κώδικα είναι η Turtle.executeChunk_. Στον Κώδικας 34 στην γραμμή 875, βλέπουμε την κλήση της συνάρτησης Turtle.checkAnswer, ο κώδικας της οποίας βρίσκεται στο Παράρτημα Β, και η οποία είναι υπεύθυνη για την αποθήκευση της λύσης του χρήστη. Για να τρέχει απλά ο κώδικας του χρήστη χωρίς να αποθηκευτεί η λύση του δημιουργήσαμε την Turtle.executeChunk2_ η οποία έχει τον ίδιο κώδικα εκτός από την γραμμή 875.

Κώδικας 34: Η συνάρτηση Turtle.executeChunk

```
851  Turtle.executeChunk_ = function() {
852      // All tasks should be complete now.  Clean up the PID list.
853      Turtle.pidList.length = 0;
854      Turtle.pause = 0;
855      var go;
856      do {
857          try {
858              go = Turtle.interpreter.step();
859          } catch (e) {
860              // User error, terminate in shame.
861              alert(e);
862              go = false;
863          }
864          if (go && Turtle.pause) {
865              // The last executed command requested a pause.
866              go = false;
867              Turtle.pidList.push(
868                  setTimeout(Turtle.executeChunk_, Turtle.pause));
869          }
870      } while (go);
871      // Wrap up if complete.
872      if (!Turtle.pause) {
873          document.getElementById('spinner').style.visibility = 'hidden';
874          BlocklyInterface.workspace.highlightBlock(null);
875          Turtle.checkAnswer();
876          // Image complete; allow the user to submit this image to gallery.
877          // Turtle.canSubmit = true;
878      }
879  };
```

Στη Ζωγραφική συναντάμε και πάλι τις συναρτήσεις execute και play που μοιράζονται τον ίδιο κώδικα, με την μόνη διαφορά ότι η Turtle.execute καλεί την συνάρτηση Turtle.executeChunk_ (Κώδικας 35, γραμμή 824) ενώ η Turlte.play καλεί την συνάρτηση Turtle.executeChunk2_ . Οι συναρτήσεις αυτές δημιουργούν ένα στιγμιότυπο του JS Interpreter όπως βλέπουμε στην γραμμή 823 με παράμετρο τον κώδικα του χρήστη και την συνάρτηση initInterpreter, αλλά ο κώδικας εκτελείται στην executeChunk με την χρήση της Turtle.interpreter.step.

Κώδικας 35: Η συνάρτηση Turtle.execute

```
811  Turtle.execute = function() {
812      if (!('Interpreter' in window)) {
813          // Interpreter lazy loads and hasn't arrived yet.  Try again later.
814          setTimeout(Turtle.execute, 250);
815          return;
816      }
817
818      Turtle.reset();
819      Blockly.selected && Blockly.selected.unselect();
820      var code = BlocklyInterface.getJsCode();
821      BlocklyInterface.executedJsCode = code;
822      BlocklyInterface.executedCode = BlocklyInterface.getCode();
823      Turtle.interpreter = new Interpreter(code, Turtle.initInterpreter);
824      Turtle.pidList.push(setTimeout(Turtle.executeChunk_, 100));
825  };
```

5.5.2. Πλήκτρο Υποβολή και τροποποίηση των Παίξε και Επαναφορά

Τα πλήκτρα «Παίξε» και «Επαναφορά» ήταν στον αρχικό κώδικα των Blockly Games και ελέγχονται από τις συναρτήσεις runButtonClick και resetButtonClick αντίστοιχα. Στον κώδικα προστέθηκε το πλήκτρο «Υποβολή» που ελέγχεται από την συνάρτηση submitButtonClick. Αρχικά δημιουργήθηκε ένα πλήκτρο με την ετικέτα button και id submitButton στα maze.soy και turtle.soy. Έπειτα συνδέθηκε το πλήκτρο με την συνάρτηση ελέγχου με την χρήση του κώδικα

```
BlocklyGames.bindClick('submitButton', Maze.submitButtonClick);
```

Στον Κώδικας 36 έχουμε τη βασική λειτουργικότητα του πλήκτρου. Παρατηρούμε ότι με τα getElementById επιλέγονται τα 3 πλήκτρα και στις γραμμές 1254-1256 απαγορεύουμε να εμφανίζονται κατά την διάρκεια που εκτελείται η συνάρτηση, με αυτόν τον τρόπο ο χρήστης δεν μπορεί να ακυρώσει την υποβολή πατώντας κάποιο άλλο πλήκτρο. Στην γραμμή 1259 καλείται η Maze.execute η οποία τρέχει τον κώδικα του χρήστη και τον αποθηκεύει.

Κώδικας 36: Κομμάτι από τον κώδικα της submitButtonClick

```
1247    var runButton = document.getElementById('runButton');
1248    var resetButton = document.getElementById('resetButton');
1249    var submitButton = document.getElementById('submitButton');
1250    // Ensure that Reset button is at least as wide as Run button.
1251    if (!resetButton.style.minWidth) {
1252        resetButton.style.minWidth = runButton.offsetWidth + 'px';
1253    }
1254    runButton.style.display = 'none'; //changed from none
1255    resetButton.style.display = 'none';
1256    submitButton.style.display = 'none';
1257
1258    Maze.reset(false);
1259    Maze.execute();
```

Ίδια είναι και η λειτουργικότητα της συνάρτησης στο turtle.js .

Για να μην αποθηκεύει ο χρήστης το αποτέλεσμά του τροποποιήθηκε η συνάρτηση runButtonClick ώστε να καλεί την play στα maze.js και turtle.js. Ο κώδικας και για τις 3 συναρτήσεις βρίσκεται στο Παράρτημα Β.

5.6. Ανάδραση παιχνιδιού – συνθήκες εμφάνισης μηνυμάτων

Μια από τις χαρακτηριστικότερες διαφοροποιήσεις του aMazeD από τα παιχνίδια Blockly είναι η χρήση του scaffolding με τα μηνύματα βοήθειας που εμφανίζονται στο πλαίσιο βοήθειας κάθε επιπέδου, στην αρχή του επιπέδου, όταν ο χρήστης πατήσει το πλήκτρο «Επαναφορά» και όταν ο χρήστης ολοκληρώσει το επίπεδο.

5.6.1. Οδηγίες επιπέδων για Λαβύρινθο και Ζωγραφική

Το πρώτο κομμάτι της βοήθειας είναι οι οδηγίες στην αρχή του κάθε επιπέδου που εμφανίζονται στην γραμμή οδηγιών. Για τον λόγο αυτό δημιουργήθηκε στα maze.soy και turtle.soy ένα div με το αναγνωριστικό header2 στο οποίο περιέχεται με ετικέτα παραγράφου το πλαίσιο των οδηγιών με αναγνωριστικό header-help. Οι οδηγίες που θα εμφανίζονται σε κάθε επίπεδο βρίσκονται επίσης στα αρχεία αυτά, ικανοποιούν τις συνθήκες των μηνυμάτων που αναλύθηκαν στην παράγραφο 5.3.3 και έχουν αναγνωριστικό dialogHelpLevelA, όπου A είναι ένας αριθμός από το 1 μέχρι το 10 που δείχνει το επίπεδο στο οποίο θα εμφανιστεί η συγκεκριμένη οδηγία.

Κώδικας 37: Δημιουργία πλαισίου οδηγιών (74-77) και παράδειγμα οδηγιών για το επίπεδο 7(339-343) στο αρχείο turtle.soy

```
74  <div id="header2">
75  <h4 id='help-ins'>{msg meaning="Maze.headerHelp" desc="callout - Instruction in header"}Level
    instructions:{/msg}</h4>
76  <p class="header-help"></p>
77  </div>

339 {if $ij.level == 7} // αρχή if επιλογή επιπέδου για οδηγίες
340  // οδηγίες επιπέδου 7
341  <div id="dialogHelpLevel7" class="dialogHiddenContent">
342      {msg meaning="Turtle.helpLevel7" desc="callout - Introduction to level 7."}Correct the blocks
    in order to draw a square with side 100 pixels.{/msg}
343   </div>
```

Ο κώδικας JavaScript που ελέγχει ποια οδηγία θα εμφανιστεί σε ποιο επίπεδο βρίσκεται στα αρχεία maze.js και turtle.js. Στον κώδικα περιέχεται ένα if που ελέγχει αν το τρέχον επίπεδο είναι ίσο με 1,2,…,10 και ανάλογα με το πιο επίπεδο έχουμε θα πάρουμε και το ανάλογο dialogHelpLevel, για παράδειγμα αν το επίπεδο είναι το 6 θα φορτωθεί το dialogHelpLevel6. Έπειτα έχουμε ένα φωλιασμένο if που ελέγχει αν το τρέχον επίπεδο έχει ολοκληρωθεί (δηλαδή αν είναι αποθηκευμένο τοπικά) και αν δεν είναι επιλέγουμε με το querySelector το πλαίσιο οδηγιών και με το getElementById το κατάλληλο κείμενο οδηγιών και το εμφανίζουμε. Στα επίπεδα 6 και 10 σε περίπτωση που το επίπεδο έχει ολοκληρωθεί

υπάρχει εναλλακτική οδηγία που ανακατευθύνει τον χρήστη στην σελίδα της Ζωγραφικής και στην σελίδα των αποτελεσμάτων αντίστοιχα.

Κώδικας 38: Επιλογή και εμφάνιση οδηγιών για το επίπεδο 6

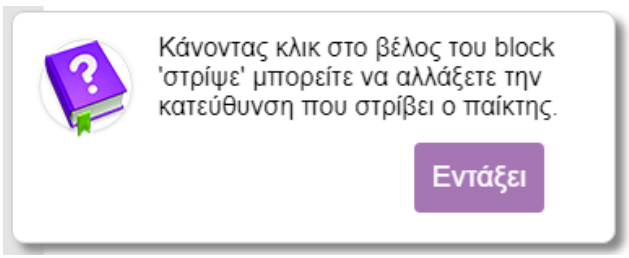
```
860     if (BlocklyGames.LEVEL == 6) {
861         if (!BlocklyGames.loadFromLocalStorage(BlocklyGames.NAME,
862                 BlocklyGames.LEVEL)) {
863             document.querySelector(".header-help").textContent = document.getElementById
                ('dialogHelpLevel6').textContent;
864         } else {
865             // //περίμενε 1 δευτερόλεπτα και άλλαξε level
866             setTimeout(BlocklyInterface.turtlePage, 1000);
867         }
868     }
```

5.6.2. Ανατροφοδότηση παίκτη

Η δεύτερη βοήθεια που προσφέρεται στον χρήστη του aMazeD είναι με την μορφή αναδυόμενων μηνυμάτων βοήθειας που τον ενημερώνουν για την λειτουργικότητα των μπλοκ ή του δίνουν μια επιπρόσθετη πληροφορία για την λογική του επιπέδου που θα τον οδηγήσει στην επιτυχή ολοκλήρωση του. Τα μηνύματα αυτά εμφανίζονται κυρίως αφού ο χρήστης επαναφέρει τον χαρακτήρα του ή το πινέλο στην αρχική θέση με το πλήκτρο «Επαναφορά» αφού έχει εκτελέσει τον κώδικά του με το πλήκτρο «Παίξε» και η λύση του δεν ήταν σωστή. Επιπλέον τα επίπεδα 1 και 2 ξεκινάνε με μήνυμα βοήθειας πριν ο χρήστης δοκιμάσει τον κώδικά του.

Τα μηνύματα αυτά παρουσιάστηκαν στο κεφάλαιο 4 και το πλαίσιο που εμφανίζονται φαίνεται στην Εικόνα 47. Το αναδυόμενο μήνυμα αποτελείται από ένα πλαίσιο με ένα εικονίδιο στα αριστερά που δημιουργήθηκε με την βοήθεια του Gimp χρησιμοποιώντας το δωρεάν εικονίδιο που πάρθηκε από την πηγή https://www.freeiconspng.com/img/17009, το μήνυμα βοήθειας που υπάρχει στο maze.soy ή turtle.soy και το πλήκτρο «Εντάξει» που κλείνει το αναδυόμενο παράθυρο και υπάρχει στο πρότυπο BlocklyGames.soy.ok .

Εικόνα 47: Αναδυόμενο μήνυμα βοήθειας στο επίπεδο 1.



Κάθε ένα από αυτά τα μηνύματα ορίζεται σε ξεχωριστό div με κλάση dialogHiddenContent και αναγνωριστικό της μορφής dialogHelpPlay ακολουθούμενο από δυο νούμερα, το πρώτο δείχνει τον αριθμό του επιπέδου. Για παράδειγμα στον Κώδικας 39 βλέπουμε το

περιεχόμενο του αναδυόμενου μηνύματος του επιπέδου 2. Παρατηρούμε ότι για να εμφανιστεί το πλήκτρο «Εντάξει» πρέπει πρώτα να κληθεί το πρότυπο BlocklyGames.soy.ok στην γραμμή 180. Η δομή του πλαισίου του μηνύματος δίνεται με τη βοήθεια πίνακα με 3 στήλες (ετικέτα <td>) στην πρώτη βρίσκεται το εικονίδιο βοήθειας, η δεύτερη είναι για οπτικούς λόγους και στην τρίτη βρίσκεται το μήνυμα και το πλήκτρο για το κλείσιμό του.

Κώδικας 39: Αναδυόμενο μήνυμα Επιπέδου 2 στο maze.soy

```
175     <div id="dialogHelpPlay22" class="dialogHiddenContent">
176      <table><tr><td valign="top">
177        <img src="maze/help.png">
178        </td><td> </td><td valign="top">
179        {msg meaning="Maze.helpPlay22" desc="callout - Help message when a
           player uses play button twice."}Think how many steps you must do
           before turning left? How many steps you do now?{/msg}
180        {call BlocklyGames.soy.ok /}
181      </td></tr>
182      </table>
183    </div>
```

Η υλοποίηση της λογικής εμφάνισης των μηνυμάτων του Λαβυρίνθου γίνεται στο maze.js με τις συναρτήσεις Maze.introHelp για τα εισαγωγικά μηνύματα στα επίπεδα 1 και 2 και την Maze.levelHelp για τα υπόλοιπα μηνύματα. Η πρώτη καλείται στην Maze.init μέσα σε μια setTimeout συνάρτηση μετά από 2 δευτερόλεπτα, επομένως εκτελείται από την αρχή του επιπέδου με καθυστέρηση 2΄΄, ενώ η δεύτερη καλείται μέσα στην Maze.resetButtonClick, επομένως έχει δυνατότητα εκτέλεσης μόνο με την χρήση του πλήκτρου «Επαναφορά». Οι συναρτήσεις αυτές έχουν 5 μεταβλητές τις:

- rtl: τύπου Boolean που ελέγχει αν η γλώσσα γραφής είναι από τα δεξιά στα αριστερά (στην περίπτωση του aMazeD είναι πάντα false).
- toolbar = BlocklyInterface.workspace.flyout_.workspace_.getTopBlocks(true): που επιστρέφει τα μπλοκ που υπάρχουν στην εργαλειοθήκη σε έναν πίνακα.
- content: με αρχική τιμή null και στην οποία θα αποθηκευτεί το κείμενο βοήθειας.
- origin: με αρχική τιμή null η οποία ελέγχει από που θα αναδυθεί το μήνυμα στο παιχνίδι.
- style: με αρχική τιμή null και με παραμέτρους width, top, left που ελέγχουν το πλάτος του παραθύρου του μηνύματος, την απόσταση του από το πάνω και από το αριστερό άκρο του παραθύρου.

Η λογική των μηνυμάτων ελέγχεται με συνθήκες if για παράδειγμα στον Κώδικας 40 βλέπουμε τον κώδικα για την εμφάνιση του μηνύματος dialogHelpPlay22, που είχαμε παρουσιάσει και στον Κώδικας 39. Το μήνυμα θα εμφανιστεί αν το αποτέλεσμα της εκτέλεσης του κώδικα δεν είναι σωστό (γραμμή 1005, Κώδικας 40) και αν ο χρήστης έχει

πατήσει το πλήκτρο Play τουλάχιστον μια φορά (γραμμή 1006). Τότε αποθηκεύεται στο content το μήνυμα με id dialogHelpPlay22 και ορίζεται το στιλ του μηνύματος στην γραμμή 1008 το οποίο αλλάζει σε ορισμένα μηνύματα ανάλογα με το μέγεθός τους. Στην γραμμή 1009 το αποτέλεσμα της rtl είναι ψευδές άρα εκτελείται η δεύτερη εντολή (το left) και στην γραμμή 1010 βλέπουμε ότι το αναδυόμενο παράθυρο θα εμφανίζεται από το toolbar[1] δηλαδή το δεύτερο μπλοκ της εργαλειοθήκης.

Κώδικας 40: Απόσπασμα από την συνάρτηση Maze.levelHelp

```
1003      } else if (BlocklyGames.LEVEL == 2) {
1004        //αν το κουμπί play πατήθηκε 2 φορές και πάνω εμφάνισε μήνυμα
1005        if (Maze.result != Maze.ResultType.UNSET) {
1006          if (Maze.levelData[1].countPlay > 0) {
1007            content = document.getElementById('dialogHelpPlay22');
1008            style = { 'width': '350px', 'top': '400px' };
1009            style[rtl ? 'right' : 'left'] = '600px';
1010            origin = toolbar[1].getSvgRoot();
1011          }
1012        }
```

Αφού ολοκληρωθούν οι συνθήκες στις δυο συναρτήσεις γίνεται η εμφάνιση του μηνύματος βοήθειας με την χρήση της συνάρτησης BlocklyDialogs.showDialog που παίρνει ως παραμέτρους τις μεταβλητές που ορίσαμε πριν και εμφανίζει το μήνυμα στο πλαίσιο που δημιουργείται με την βοήθεια του προτύπου BlocklyGames.soy.dialog.

Τα επίπεδα της Ζωγραφικής έχουν λίγο διαφορετική δομή γιατί υπάρχει ήδη στα δεξιά της γραμμής πλοήγησης το πλήκτρο Βοήθεια. Αυτό ουσιαστικά είναι ένα div στο turtle.soy που περιέχει τα μηνύματα βοήθειας ανάλογα με το επίπεδο του aMazeD (γραμμή 234, Κώδικας 41) και έχει το αναγνωριστικό help. Η συνάρτηση που υλοποιεί την λειτουργικότητα των μηνυμάτων βοήθειας είναι η Turtle.showHelp και βρίσκεται στο turtle.js.

Κώδικας 41: Μήνυμα βοήθειας στο turtle.soy για το επίπεδο 7

```
232   <div id="help" class="dialogHiddenContent">
233     <div style="padding-bottom: 0.7ex">
234     {if $ij.level == 7}
235
236      <table><tr><td valign="top">
237        <img src="maze/help.png">
238        </td><td> </td><td valign="top">
239        {msg meaning="Turtle.helpLevel71" desc="callout - Help message when a
           player uses play button once."}Try to turn in the same direction!{/
           msg}
240       </td></tr>
241       </table>
```

Στον παρακάτω κώδικα παρατηρούμε ότι πάλι ορίζονται οι παράμετροι width, left, top των μηνυμάτων αλλά το περιεχόμενο του μηνύματος αποθηκεύεται με την βοήθεια του help,

στο οποίο υπάρχουν οι συνθήκες για τα επίπεδα. Η εμφάνιση του μηνύματος γίνεται και πάλι με την χρήση της BlocklyDialogs.showDialog όπως και στον Λαβύρινθο. Η κλήση της συνάρτησης Turtle.showHelp γίνεται στο Turtle.resetButtonClick και ακολουθεί την λογική του Λαβυρίνθου, επιπλέον το αναδυόμενο μήνυμα θα εμφανιστεί πατώντας το πλήκτρο Βοήθεια με τη χρήση του κώδικα

```
BlocklyGames.bindClick('helpButton', Turtle.showHelp);
```

στην Turtle.init().

Κώδικας 42: Η συνάρτηση Turtle.showHelp που εμφανίζει τα μηνύματα βοήθειας στα επίπεδα της Ζωγραφικής

```
460  Turtle.showHelp = function() {
461      var help = document.getElementById('help');
462      var button = document.getElementById('helpButton');
463      var style = {
464          width: '30%',
465          left: '40%',
466          top: '350px'
467      };
468
469      if (BlocklyGames.LEVEL >= 7 && BlocklyGames.LEVEL <= 10) {
470          //μήνυμα μετά το πρώτο play
471          BlocklyDialogs.showDialog(help, button, true, true, style, Turtle.
             hideHelp);
472          BlocklyDialogs.startDialogKeyDown();
473
474      }
```

Τέλος στα επίπεδα της Ζωγραφικής υπάρχει ένα ακόμα μήνυμα βοήθειας για τις κατηγορίες που εμφανίζονται στην εργαλειοθήκη, αυτό γίνεται μέσω της συνάρτησης Turtle.showCategoryHelp, η οποία υπήρχε και στα παιχνίδια Blockly και υιοθετήθηκε στο aMazeD.

5.6.3. Αναδυόμενα μηνύματα ολοκλήρωσης επιπέδου

Με την ολοκλήρωση του επιπέδου ο χρήστης ενημερώνεται για το αν η προσπάθεια του είναι Επιτυχής ή όχι και βλέπει τον κώδικα των μπλοκ του σε κάθε περίπτωση. Το μήνυμα για την επιτυχή ολοκλήρωση του επιπέδου υπήρχε ήδη στον κώδικα των παιχνιδιών Blockly, για το aMazeD το τροποποιήσαμε για να εμφανίζεται και στην περίπτωση που ο χρήστης δεν ολοκληρώνει το επίπεδο αλλά η λύση του είναι λάθος.

Αρχικά δημιουργήθηκε το πρότυπο .doneDialog2 στο BlocklyGames.soy με τον παρακάτω κώδικα:

```
194  {template .doneDialog2 private="true"}
195    <div id="dialogDone2" class="dialogHiddenContent">
196      <div style="font-size: large; margin: 1em; color: red;">{msg
       meaning="Games.submitWrong" desc="alert - This is displayed when the user
       submits unsuccessful result of the level."}You didn't solve the level.{/
       msg}</div>
197      <div id="dialogLinesText2" style="font-size: large; margin: 1em;"></div>
198      <pre id="containerCode2"></pre>
199      <div id="dialogDoneText2" style="font-size: large; margin: 1em;"></div>
200      <div id="dialogDoneButtons2" class="farSide" style="padding: 1ex 3ex 0">
201        <button id="doneOk2" class="secondary">
202          {{msg meaning="Games.dialogOk" desc="IBID"}}OK{{/msg}}
203        </button>
204      </div>
205    </div>
206  {/template}
```

Έγινε κλήση του προτύπου στα πρότυπα maze.soy και turtle.soy, με την εντολή:

```
{call BlocklyGames.soy.doneDialog2 /}
```

Δημιουργήθηκε στο αρχείο lib-dialogs.js η συνάρτηση BlocklyDialogs.endOfLevel, αντίστοιχη της BlocklyDialogs.congratulations που ενημερώνει τον χρήστη για την σωστή επίλυση του επιπέδου, ο κώδικας της οποίας υπάρχει στο παράρτημα Β. Τέλος προσθέσαμε την κλήση της συνάρτησης αυτής στις συναρτήσεις Maze.animate και Turtle.checkAnswer όπως φαίνεται στα στιγμιότυπα που ακολουθούν.

Κώδικας 43: Κλήση των congratulations και endOfLevel στην Maze.animate

```
1547          case 'end':
1548              BlocklyInterface.saveToLocalStorage();
1549              setTimeout(BlocklyDialogs.endOfLevel, 1000);
1550              break;
1551          case 'finish':
1552              Maze.scheduleFinish(true);
1553              BlocklyInterface.saveToLocalStorage();
1554              setTimeout(BlocklyDialogs.congratulations, 1000);
1555      }
```

Στο στιγμιότυπο 43 παρατηρούμε ότι έχουμε χωρίσει τα δυνατά αποτελέσματα σε δυο περιπτώσεις: το end που τερματίζει το επίπεδο αλλά η λύση είναι λάθος και καλεί την endOfLevel και το finish που τερματίζει το επίπεδο και η λύση είναι σωστή και καλεί την congratulations. Αντίστοιχα στο στιγμιότυπο Κώδικας 44 βλέπουμε την συνθήκη ελέγχου της λύσης του χρήστη στην γραμμή 1075. Αν η λύση είναι σωστή καλείται η congratulations (γραμμή 1083), ενώ αν είναι λανθασμένη καλείται η endOfLevel (γραμμή 1092).

Κώδικας 44: Κλήση των congratulations και endOfLevel στην Turtle.checkAnswer

```
1075        if (Turtle.isCorrect(delta)) {
1076            //αποθήκευσε δεδομένα επιτυχίας
1077            document.getElementById("msgSuccess").style.display = 'inline';
1078            Turtle.levelData[BlocklyGames.LEVEL - 7].score = 10 * BlocklyGames.
                LEVEL;
1079            Turtle.levelData[BlocklyGames.LEVEL - 7].result = 'Success';
1080            BlocklyInterface.saveToLocalStorage();
1081            Turtle.saveScore();
1082            BlocklyInterface.workspace.getAudioManager().play('win', 0.5);
1083            BlocklyDialogs.congratulations();
1084        } else if (!Turtle.isCorrect(delta)) {
1085            //αν δεν είναι επιτυχημένη η προσπάθεια αλλά πατήσει submit
1086            //αποθήκευσε δεδομένα αποτυχίας
1087            document.getElementById("msgFail").style.display = 'inline';
1088            Turtle.levelData[BlocklyGames.LEVEL - 7].score = 0;
1089            Turtle.levelData[BlocklyGames.LEVEL - 7].result = 'Failure';
1090            BlocklyInterface.saveToLocalStorage();
1091            Turtle.saveScore();
1092            BlocklyDialogs.endOfLevel();
1093        }
1094    };
```

5.7. Δεδομένα επιπέδων και εμφάνιση αποτελεσμάτων

5.7.1. Αποτελέσματα ανά επίπεδο

Τα αποτελέσματα ανά επίπεδο εμφανίζονται στο πλαίσιο αποτελεσμάτων, όπως ήδη αναφέραμε στο Κεφάλαιο 4, ο κώδικας με τις κατάλληλες συναρτήσεις βρίσκεται στα αρχεία Maze.soy, Turtle.soy, maze.js και turtle.js και είναι ίδιος και για τις δυο κατηγορίες. Αρχικά δημιουργήθηκε μια ενότητα στα Maze.soy και Turtle.soy με την ετικέτα και αναγνωριστικό id= ‘outcome’ όπως φαίνεται στον Κώδικας 45.

Κώδικας 45: Πλαίσιο αποτελεσμάτων σε Maze.soy

```
99  <div id='outcome'>
100 <p class='outcome__label'>============ {msg meaning="Maze.
    outcomeResult" desc="callout"}Result{/msg} ============</p>
101 <p class='outcome__label'> ▶ {msg meaning="Maze.outcomePlay"
    desc="callout"}Play button count:{/msg} <span class='outcome__value
    outcome__value--play'>0</span></p>
102 <p class='outcome__label'>---------------------------------------</p>
103 <p class='outcome__label'>{msg meaning="Maze.outcomeRLevel"
    desc="callout"}Level outcome: {/msg}<span id='msgSuccess'> {msg
    meaning="Maze.outcomeSuccess" desc="callout"}Success{/msg}</
    span><span id='msgFail'> {msg meaning="Maze.outcomeFail"
    desc="callout"}Fail{/msg}</span></p>
104 <p class='outcome__label'>---------------------------------------</p>
105   <p class='outcome__label'>💯 {msg meaning="Maze.levelScore"
      desc="callout"}Level score: {/msg}<span class='outcome__value
      outcome__value--score'> </span></p>
106   <p class='outcome__label'>🏆 {msg meaning="Maze.totalScore"
      desc="callout"}Total score: {/msg}<span class='outcome__value
      outcome__value--total__score'>0</span></p>
107 </div>
```

Όλα τα παιδιά του outcome είναι στοιχεία παραγράφου και μορφοποιούνται με την χρήση της κλάσης ‘outcome__label’ . Τα αποτελέσματα που εμφανίζονται στο πλαίσιο είναι :

- Πλήθος ‘Παίξε’
- Αποτελέσματα Επιπέδου
- Βαθμολογία Επιπέδου
- Συνολική Βαθμολογία

Για κάθε ένα από αυτά δημιουργείται ένα στοιχείο παραγράφου με ετικέτα … και μια κλάση ή ένα id για την επιλογή και την μορφοποίηση του στοιχείου όπως είναι η outcome__value--play στην γραμμή 101, ενώ μέσω της ετικέτας msg είναι δυνατή η μετάφραση του στοιχείου στα αρχεία json όπως είδαμε στην παράγραφο 5.3.1 .

Για την αποθήκευση των αποτελεσμάτων δημιουργήθηκαν δυο πίνακες ο Maze.levelData και ο Turtle.levelData στα αρχεία maze.js και turtle.js. Κάθε πίνακας έχει ως στοιχεία του αντικείμενα της JavaScript, κάθε αντικείμενο αντιστοιχεί σε ένα επίπεδο με ιδιότητες αντικειμένου

- name: αλφαριθμητικό για το όνομα του επιπέδου,
- countPlay: αριθμός που δίνει πόσα «Παίξε» έχουν χρησιμοποιηθεί στο επίπεδο
- time: αποθηκεύει το χρόνο του χρονομέτρου μέχρι την στιγμή που πατήθηκε το πλήκτρο «Υποβολή»
- result: αλφαριθμητικό που δέχεται δυο τιμές «Αποτυχία» ή «Επιτυχία» ανάλογα με το αποτέλεσμα του επιπέδου
- score: αριθμός που δίνει το σκορ του επιπέδου (αριθμός επιπέδου * 10) σε περίπτωση επιτυχίας ή παραμένει μηδέν σε περίπτωση αποτυχίας

Τα αντικείμενα ξεκινούν με προκαθορισμένες τιμές όπως φαίνεται στον Κώδικας 46.

Κώδικας 46: Πίνακας δεδομένων Ζωγραφικής

```
66  Turtle.levelData = [{
67          name: "Επίπεδο 7",
68          countPlay: 0,
69          time: "0",
70          result: "",
71          score: 0,
72      },
73      {
74          name: "Επίπεδο 8",
75          countPlay: 0,
76          time: "0",
77          result: "",
78          score: 0,
79      },
80      {
81          name: "Επίπεδο 9",
82          countPlay: 0,
83          time: "0",
84          result: "",
85          score: 0,
86      },
87      {
88          name: "Επίπεδο 10",
89          countPlay: 0,
90          time: "0",
91          result: "",
92          score: 0,
93      },
94  ];
```

Για το countPlay στις συναρτήσεις Maze.runButtonClick και Turtle.runButtonClick προσθέσαμε τον Κώδικας 47, ο οποίος αλλάζει μόνο όσον αφορά το πρόθεμα Maze στην Ζωγραφική, πιο συγκεκριμένα τα προθέματα Maze γίνονται Turtle.

Κώδικας 47: countPlay στον Λαβύρινθο

```
1210    Maze.levelData[BlocklyGames.LEVEL - 1].
        countPlay += 1;
1211    document.querySelector(".
        outcome__value--play").textContent = Maze.
        levelData[BlocklyGames.LEVEL - 1].
        countPlay;
```

Κάθε φορά που εκτελείται η συνάρτηση runButtonClick αυξάνεται η μεταβλητή countPlay και με την χρήση του document.querySelector επιλέγεται μέσω της κλάσης το επιθυμητό στοιχείο παραγράφου που αντιστοιχεί στο πλήθος των «Παίξε» και αντικαθιστούμε την προκαθορισμένη τιμή του με την μεταβλητή countPlay στο πλαίσιο των αποτελεσμάτων.

Το αποτέλεσμα του επιπέδου όπως και η βαθμολογία του γίνεται στις συναρτήσεις Maze.execute και Maze.play όπως είδαμε στην παράγραφο 5.5 στο στιγμιότυπο Κώδικας 33, εκεί εκχωρείται η βαθμολογία του επιπέδου στην ιδιότητα score του αντικειμένου levelData και η τιμή Επιτυχία/Αποτυχία στην ιδιότητα result του ίδιου αντικειμένου. Για την αποθήκευση της βαθμολογίας του κάθε επιπέδου δημιουργήσαμε τις συναρτήσεις Maze.saveScore και Turtle.saveScore στα αρχεία maze.js και turtle.js αντίστοιχα. Ο

κώδικας και για τις δυο συναρτήσεις είναι ίδιος (αλλάζουν μόνο τα Maze και Turtle) και παρουσιάζεται στο στιγμιότυπο Κώδικας 48.

Κώδικας 48: Συνάρτηση saveScore στο turtle.js

```
1034  Turtle.saveScore = function() {
1035
1036      var lname = "maze_level" + BlocklyGames.LEVEL;
1037      window.localStorage.setItem(lname, JSON.stringify(Turtle.levelData
          [BlocklyGames.LEVEL - 7]));
1038
1039      document.querySelector(".outcome__value--score").textContent =
          Turtle.levelData[BlocklyGames.LEVEL - 7].score;
1040      if (window.localStorage.getItem('total-score')) {
1041          totalScore = totalScore + Turtle.levelData[BlocklyGames.LEVEL
              - 7].score;
1042          document.querySelector('.outcome__value--total__score').
              textContent = totalScore;
1043          window.localStorage.setItem('total-score', totalScore);
1044      }
1045
1046  }
```

Στην γραμμή 1036 δημιουργείται η τοπική μεταβλητή lname που θα χρησιμοποιηθεί για να αποθηκευτεί στο LocalStorage του φυλλομετρητή ο αριθμός του επιπέδου από maze_level1 μέχρι maze_level10[16]. Στη συνέχεια με την μέθοδο JSON.stringify περνάμε με την μορφή αλφαριθμητικού τα δεδομένα του πίνακα Turtle.levelData (ή Maze.levelData για τον Λαβύρινθο) για το συγκεκριμένο επίπεδο, παρατηρούμε ότι για να πάρουμε τα σωστά δεδομένα στο αρχείο turtle πρέπει να αφαιρέσουμε από το τρέχον επίπεδο τον αριθμό 7, αφού ο πίνακας levelData έχει 4 στοιχεία ξεκινά από την θέση 0 για το επίπεδο 7 και έχει τελευταίο αντικείμενο στη θέση 3 για το επίπεδο 10.

Στην γραμμή 1039 επιλέγουμε με την μέθοδο document.querySelector(.κλάση).textContent το κείμενο του στοιχείου με κλάση outcome__value--score και το αντικαθιστούμε με το score του επιπέδου. Στη συνέχεια στις γραμμές 1040-1043 προσθέτουμε την βαθμολογία του τρέχοντος επιπέδου στο συνολικό σκορ του χρήστη, εμφανίζουμε το συνολικό σκορ στο πλαίσιο αποτελεσμάτων αντικαθιστώντας την τιμή του στοιχείου με κλάση outcome__value--total__score και την αποθηκεύουμε στο LocalStorage του φυλλομετρητή με όνομα κλειδιού total-score. Η συνθήκη if στην γραμμή 1040 ελέγχει αν υπάρχει το στοιχείο total-score αποθηκευμένο στο LocalStorage του φυλλομετρητή και αποτελεί μια καλή πρακτική για την αποφυγή λαθών όταν δουλεύουμε με το window.LocalStorage. Η

[16] Τα επίπεδα είναι στη μορφή maze_level και δεν αλλάζουν σε turtle_level, καταλαβαίνουμε το επίπεδο της Ζωγραφικής από τον αριθμό του (επίπεδα 7-10)

κλήση της παραπάνω συνάρτησης γίνεται μέσα στην Maze.submitButtonClick για τον Λαβύρινθο και στην Turtle.checkAnswer για την Ζωγραφική.

Τέλος για την σωστή λειτουργία των παραπάνω συναρτήσεων και της χρήσης των αντικειμένων των πινάκων Maze.levelData και Turtle.levelData τοποθετήσαμε στις συναρτήσεις Maze.init και Turtle.init την αντικατάσταση των προκαθορισμένων τιμών τους με τα δεδομένα που είναι αποθηκευμένα στο LocalStorage. Κάθε φορά που θα γίνεται η έναρξη ενός επιπέδου είτε θα γίνεται refresh της σελίδας θα βλέπουμε στο πλαίσιο των αποτελεσμάτων τα δεδομένα που είναι αποθηκευμένα στο LocalStorage. Ο κώδικας στην Maze.init παρουσιάζεται στο στιγμιότυπο που ακολουθεί.

Κώδικας 49: Αρχικοποίηση αντικειμένων στον πίνακα Maze.levelData στην συνάρτηση Maze.init

```
517     var data1 = JSON.parse(window.localStorage.getItem("maze_level1"));
518     var data2 = JSON.parse(window.localStorage.getItem("maze_level2"));
519     var data3 = JSON.parse(window.localStorage.getItem("maze_level3"));
520     var data4 = JSON.parse(window.localStorage.getItem("maze_level4"));
521     var data5 = JSON.parse(window.localStorage.getItem("maze_level5"));
522     var data6 = JSON.parse(window.localStorage.getItem("maze_level6"));
523
524
525     if (data1) Maze.levelData[0] = data1;
526     if (data2) Maze.levelData[1] = data2;
527     if (data3) Maze.levelData[2] = data3;
528     if (data4) Maze.levelData[3] = data4;
529     if (data5) Maze.levelData[4] = data5;
530     if (data6) Maze.levelData[5] = data6;
```

Στις γραμμές 517-522 με την μέθοδο Json.parse, η οποία επιστρέφει το αλφαριθμητικό όρισμα σε αντικείμενο τύπου Json, περνάμε τα δεδομένα των επιπέδων 1-6 στο LocalStorage στα αντικείμενα data1-data6. Στη συνέχεια στις γραμμές 525-530 εκχωρούμε τα αντικείμενα αυτά, εφόσον υπάρχουν δεδομένα, στα αντικείμενα του πίνακα levelData και με αυτόν τον τρόπο κάθε φορά που ξεκινά ένα επίπεδο έχουμε τις σωστές τιμές στις ιδιότητες των επιπέδων.

5.7.2. Δομή των δεδομένων στο LocalStorage του φυλλομετρητή

Στον Πίνακας 11 και στην Εικόνα 48 περιγράφονται τα δεδομένα του χρήστη που είναι αποθηκευμένα στο LocalStorage μετά την ολοκλήρωση του παιχνιδιού.

Πίνακας 11: Τα ζεύγη δεδομένων (αριστερά) Κλειδί, (δεξιά) Τιμή που αποθηκεύονται στο LocalStorage για το aMazeD.

Κλειδί	Τιμή
maze_levelA	A=1,2,…,10 Αποθηκεύεται με την μορφή αλφαριθμητικού το αντικείμενο του levelData που αντιστοιχεί στο A επίπεδο.
mazeB	B=1,…,6 Αποθηκεύονται σε μορφή xml τα μπλοκ του χρήστη για το B επίπεδο.
turtleC	C=7,…10 Αποθηκεύονται τα μπλοκ του χρήστη για το C επίπεδο C=1,..,6 Αποθηκεύεται η τιμή no game, αποτελούν εικονικά επίπεδα.
maze-codeZ	Z=1,..,10 Αποθηκεύεται ένα αλφαριθμητικό με τον κώδικα του χρήστη για το Z επίπεδο σε JavaScript.

Στην Εικόνα 48 βλέπουμε την μορφή του LocalStorage αφού ο χρήστης έχει ολοκληρώσει ένα παιχνίδι. Για να κατανοηθεί το εύρος των δεδομένων που αποθηκεύονται από τις συναρτήσεις του aMazeD παρουσιάζουμε τις πληροφορίες που βλέπουμε στην Εικόνα 48 για το επίπεδο 7:

- Επίπεδο 7
- Πλήθος «Παίξε» : 0
- Χρόνος : 18 λεπτά και 57 δευτερόλεπτα
- Αποτέλεσμα: Επιτυχία
- Βαθμολογία Επιπέδου: 70
- Κώδικας χρήστη σε JavaScript:

```
moveForward (100)
turnRight (90)
moveForward (100)
turnRight (90)
moveForward (100)
turnRight (90)
moveForward (100)
```

Επιπλέον βλέπουμε ότι το συνολικό σκορ του χρήστη είναι total-score=210.

Εικόνα 48: LocalStorage του Mozilla Firefox μετά την ολοκλήρωση ενός παιχνιδιού aMazeD

Key	Value
turtle7	<xml xmlns="https://developers.google.com/blockly/xml"><block type="turtle_move_internal"><field name="DIR">moveFor
turtle6	no game
turtle5	no game
turtle4	no game
turtle3	no game
turtle2	no game
turtle1	no game
total-score	210
name	[redacted]
maze_level10	{"name":"Επίπεδο 10","dm":0,"time":"00:05","result":"Failure","hl":0}
maze_level9	{"name":"Επίπεδο 9","dm":0,"time":"19:32","result":"Success","hl":90}
maze_level8	{"name":"Επίπεδο 8","dm":1,"time":"15:14","result":"Success","hl":80}
maze_level7	{"name":"Επίπεδο 7","dm":0,"time":"18:57","result":"Success","hl":70}
maze_level6	{"name":"Επίπεδο 6","Ab":2,"time":"05:43","result":"Success","yc":60}
maze_level5	{"name":"Επίπεδο 5","Ab":3,"time":"05:19","result":"Success","yc":50}
maze_level4	{"name":"Επίπεδο 4","Ab":3,"time":"06:26","result":"Success","yc":40}
maze_level3	{"name":"Επίπεδο 3","Ab":2,"time":"06:05","result":"Success","yc":30}
maze_level2	{"name":"Επίπεδο 2","Ab":2,"time":"08:42","result":"Success","yc":20}
maze_level1	{"name":"Επίπεδο 1","Ab":2,"time":"16:32","result":"Success","yc":10}
maze-code10	for (var count2 = 0; count2 < 4; count2++) { for (var count = 0; count < 3; count++) { moveForward(50); turnRight(120); }
maze-code9	for (var count = 0; count < 3; count++) { moveForward(50); turnRight(120);}
maze-code8	for (var count = 0; count < 2; count++) { moveForward(50); turnRight(90); moveForward(100); turnRight(90);}
maze-code7	moveForward(100);turnRight(90);moveForward(100);turnRight(90);moveForward(100);turnRight(90);moveForward(100);
maze-code6	while (notDone()) { if (isPathForward()) { moveForward(); } else { if (isPathRight()) { turnRight(); } else { turnLeft(); }
maze-code5	while (notDone()) { moveForward(); if (isPathLeft()) { turnLeft(); } if (isPathRight()) { turnRight(); }}
maze-code4	while (notDone()) { moveForward(); if (isPathRight()) { turnRight(); }}
maze-code3	for (var count2 = 0; count2 < 3; count2++) { for (var count = 0; count < 4; count++) { moveForward(); } turnRight();}moveF
maze-code2	while (notDone()) { turnRight(); moveForward(); moveForward(); moveForward(); turnLeft(); moveForward();}
maze-code1	moveForward();turnLeft();moveForward();moveForward();turnRight();moveForward();
maze6	<xml xmlns="https://developers.google.com/blockly/xml"><block type="maze_forever"><statement name="DO"><block typ
maze5	<xml xmlns="https://developers.google.com/blockly/xml"><block type="maze_forever"><statement name="DO"><block typ
maze4	<xml xmlns="https://developers.google.com/blockly/xml"><block type="maze_forever"><statement name="DO"><block typ
maze3	<xml xmlns="https://developers.google.com/blockly/xml"><block type="controls_repeat_ext"><field name="TIMES">3</fiel
maze2	<xml xmlns="https://developers.google.com/blockly/xml"><block type="maze_forever"><statement name="DO"><block typ
maze1	<xml xmlns="https://developers.google.com/blockly/xml"><block type="maze_moveForward"><next><block type="maze_tu
email	[redacted]@gmail.com

5.7.3. Σελίδα αποτελεσμάτων

Για να είναι πιο ευανάγνωστα τα αποτελέσματα του παιχνιδιού δημιουργήσαμε την σελίδα των αποτελεσμάτων result.html στην οποία ανακατευθύνεται ο χρήστης με το τέλος του παιχνιδιού. Για την σελίδα αυτή όπως και για την about.html χρησιμοποιήθηκε το πλαίσιο ανάπτυξης Bootstrap, με κύριο στόχο να είναι οι ιστοσελίδες ευπροσάρμοστες (responsive) σε όλες τις διαθέσιμες συσκευές. Η δομή της σελίδας περιγράφεται στην παράγραφο 4.5.2 και ο κώδικας της result-el.html και της about-en.html βρίσκονται στο Παράρτημα Β της παρούσας εργασίας. Το αρχείο results.html περιέχει:

- Την γραμμή πλοήγησης με την φόρμα των στοιχείων στα οποία χρησιμοποιούμε τα id name, email για να έχουμε πρόσβαση στις τιμές του.
- Ένα div με id app που περιέχει την κύρια εφαρμογή των αποτελεσμάτων.

Στην εφαρμογή εμφανίζονται

- ένα συγχαρητήριο μήνυμα ακολουθούμενο από το συνολικό σκορ του χρήστη, που έχει id total-score
- ένα πλήκτρο για διαγραφή των δεδομένων του χρήστη με id clear
- ο πίνακας που περιέχει τα αποτελέσματα με id myTable, στον οποίο εμφανίζονται 10 γραμμές με τα αποτελέσματα του επιπέδου. Για να έχουμε πρόσβαση στα αποτελέσματα αυτά δώσαμε τα αναγνωριστικά: result-n, score-n, time-n, play-n, code-n, με n=1,…10 που δίνουν πρόσβαση στα κελιά της n γραμμής που αντιστοιχούν στα πεδία Αποτέλεσμα, Βαθμολογία, Χρόνος, Πλήθος play, Κώδικας JavaScript.
- ένα πλήκτρο για την αποθήκευση του πίνακα σε pdf μορφή, με αναγνωριστικό pdfExport
- ένα πλήκτρο για την αποθήκευση του πίνακα σε αρχείο Excel, με αναγνωριστικό xlsExport.

Το εικονίδιο που εμφανίζεται αριστερά του σκορ και παρουσιάζεται στην Εικόνα 49 δημιουργήθηκε με την βοήθεια του λογισμικού Gimp με την βοήθεια του εικονιδίου του Λαβυρίνθου στην αρχική σελίδα των παιχνιδιών Blockly (βλ. Πίνακας 5: Παιχνίδια Blockly).

Εικόνα 49: Εικονίδιο στα αριστερά του σκορ, στη σελίδα των Αποτελεσμάτων

Το αρχείο results.html συνδέεται με 2 αρχεία css, το results.css και το bootstrap-4.3.1.css, μέσω της ετικέτας <link> στην αρχή του κώδικα και με 4 αρχεία js τα results.js, jquery-3.3.1.min.js, popper.min.js και bootstrap-4.3.1.js μέσω της ετικέτας <script> στο τέλος του κώδικα. Τα αρχεία results.css και results.js δημιουργήθηκαν στα πλαίσια αυτής της εργασίας ενώ τα υπόλοιπα αρχεία είναι διαθέσιμα στο πλαίσιο Bootstrap και είναι απαραίτητα για την εφαρμογή του πλαισίου στην ιστοσελίδα.

Με την ανακατεύθυνση στην σελίδα ο χρήστης βλέπει μόνο την γραμμή εισαγωγής στοιχείων, αυτό επιτυγχάνεται με τον παρακάτω κώδικα στο αντίστοιχο αρχείο result.js.

Στις γραμμές 136-145 ανακτώνται τα στοιχεία που θα καταχωρήσει ο χρήστης στα κατάλληλα πλαίσια και αποθηκεύονται τοπικά. Στη γραμμή 147 ελέγχεται αν ο χρήστης έχει δώσει στοιχεία και αν η συνθήκη είναι αληθής, η ορατότητα των αποτελεσμάτων

(style.opacity[17]) που εμφανίζονται στο div με αναγνωριστικό app γίνεται 100, η προκαθορισμένη τιμή στο result.css είναι 0.

Κώδικας 50: Ανάκτηση στοιχείων φόρμας και εμφάνιση αποτελεσμάτων στην result.html

```
136   // ανάκτηση στοιχείων Φόρμας
137   document.getElementById('save-btn').addEventListener('click', function() {
138       //prevent form from submitting
139       // e.preventDefault();
140       let userName = document.getElementById('name').value;
141       let userEmail = document.getElementById('myemail').value;
142       console.log(userName, userEmail);
143       window.localStorage.setItem('name', userName);
144       window.localStorage.setItem('email', userEmail);
145   });
146
147   if (window.localStorage.getItem('name')) {
148       document.getElementById('app').style.opacity = 100;
```

Στο result.js υπάρχει ένας πίνακας levelData δέκα αντικειμένων, στα οποία θα αποθηκευτούν τα στοιχεία των επιπέδων, και υλοποιεί την λογική του Maze.levelData που είδαμε σε προηγούμενη παράγραφο. Η αποθήκευση των κατάλληλων στοιχείων γίνεται μέσα σε ένα βρόγχο for (Κώδικας 51) που θα κάνει δέκα επαναλήψεις, μια για κάθε επίπεδο. Κάθε επανάληψη αντιστοιχεί στην ανάκτηση και την εμφάνιση των δεδομένων στο επίπεδο. Οι ενέργειες που εκτελούνται στον βρόγχο για κάθε επίπεδο είναι:

- ανακτά από το LocalStorage την τιμή του maze_level και την αποθηκεύει με την χρήση της JSON.parse στην τοπική μεταβλητή data,
- ανακτά την τιμή του κλειδιού maze-code και την αποθηκεύει στην μεταβλητή code,
- αν η data δεν είναι null αποθηκεύει το JSON ως αντικείμενο στον πίνακα levelData και επιλέγοντας το κατάλληλο κελί του myTable, εμφανίζει τις ιδιότητες του αντικειμένου στον πίνακα
- αν η code δεν είναι null αποθηκεύει τον κώδικα στην ιδιότητα code του αντικειμένου και τον εμφανίζει στο κατάλληλο κελί του myTable.

Εκτός του βρόγχου ανακτάται και εμφανίζεται το τελικό σκορ του χρήστη (Κώδικας 51, γραμμές 120-124).

[17] παίρνει τιμές στο διάστημα 0-100 με το 0 να αντιστοιχεί σε καθόλου και το 100 σε πλήρη ορατότητα

Κώδικας 51: Ανάκτηση στοιχείων από LocalStorage και εμφάνισή τους στον πίνακα Αποτελεσμάτων

```
81   for (let i = 0; i < levelData.length; i++) {
82       let temp = i + 1;
83       let name = "maze_level" + temp;
84       let codeLevel = 'maze-code' + temp;
85
86       let playid = "play-" + temp;
87       let timeid = "time-" + temp;
88       let resultid = "result-" + temp;
89       let scoreid = "score-" + temp;
90       let result = '';
91       let codeid = 'code-' + temp;
92       let data = JSON.parse(window.localStorage.getItem(name));
93       console.log(data);
94       let code = window.localStorage.getItem(codeLevel);
95       if (data) levelData[i] = data;
96       if (temp <= 6) {
97           document.getElementById(scoreid).textContent = levelData[i].yc;
98           document.getElementById(playid).textContent = levelData[i].Ab;
99       } else {
100          document.getElementById(scoreid).textContent = levelData[i].hl;
101          document.getElementById(playid).textContent = levelData[i].dm;
102      }
103      console.log(levelData[i].Vf);
104      document.getElementById(timeid).textContent = levelData[i].time;
105      if (levelData[i].result == 'Failure') {
106          result = 'Αποτυχία';
107      } else {
108          result = 'Επιτυχία';
109      }
110      document.getElementById(resultid).textContent = result;
111
112
113      // για να εμφανίσει σωστά τον κώδικα!!!
114      if (code) {
115          levelData[i].code = code;
116          document.getElementById(codeid).innerText = code;
117      }
118  }

120  // εμφάνισε total-score
121  const totalScore = window.localStorage.getItem("total-score");
122  document.getElementById(
123      "total-score"
124  ).textContent = totalScore;
```

Η αποθήκευση των δεδομένων του χρήστη σε pdf ή σε xls γίνεται με τις συναρτήσεις Export() και exportTableToExcel() ο κώδικας των οποίων βρίσκεται στο Παράρτημα Β. Για να περιλαμβάνει το αποθηκευμένο αρχείο τα στοιχεία του χρήστη, ημερομηνία και ώρα που ολοκληρώθηκε το παιχνίδι και το τελικό σκορ, δημιουργήθηκε μια λεζάντα στον πίνακα αποτελεσμάτων με αναγνωριστικό caption—text.

Κώδικας 52: Δημιουργία και εμφάνιση ονόματος, email, σκορ, ημερομηνίας και ώρας παιχνιδιού του χρήστη

```
185  // ημερομηνία :
186  const now = new Date();
187  const options = {
188      hour: 'numeric',
189      minute: 'numeric',
190      day: 'numeric',
191      month: 'numeric',
192      year: 'numeric',
193      // weekday: 'long',
194  };
195
196  const labelDate = new Intl.DateTimeFormat(
197      'el-GR',
198      options
199  ).format(now);
200
201  //εμφάνισε όνομα και email στην πρώτη γραμμή του πίνακα
202
203  const getName = window.localStorage.getItem('name');
204  const getEmail = window.localStorage.getItem('email');
205  const capTxt = 'Στοιχεία χρήστη --> Ονοματεπώνυμο: ' + getName + ' , e-mail: ' + getEmail + ' ,
     Score: ' + totalScore + ' , Ημερ/νια: ' + labelDate;
206  console.log(capTxt);
207  document.getElementById('caption--text').innerHTML = capTxt;
```

Στις γραμμές 186-199 του Κώδικας 52 δημιουργούμε την ημερομηνία και επιλέγουμε την ελληνική μορφοποίηση της εμφάνισής της με την χρήση της συνάρτησης Intl.DateTimeFormat . Στις γραμμές 203-207 με την χρήση των μεταβλητών getName, getEmail, totalScore ανακτούμε από το LocalStorage τα δεδομένα του χρήστη και το σκορ του και τα αποθηκεύουμε μαζί με την ημερομηνία (labelDate) στην μεταβλητή capTxt που περιέχει πλέον ένα αλφαριθμητικό με τα στοιχεία που θα εμφανιστούν στη λεζάντα του πίνακα. Τέλος εμφανίζουμε την λεζάντα στο στοιχείο με ετικέτα <caption> και αναγνωριστικό caption—text, που έχει δημιουργηθεί στην results.html.

Ο χρήστης έχει δικαίωμα να διαγράψει όλα τα δεδομένα που αποθηκεύτηκαν τοπικά με την χρήση του πλήκτρου Νέο Παιχνίδι. Το πλήκτρο έχει το αναγνωριστικό clear και όταν ο χρήστης κάνει κλικ σε αυτό καλείται η συνάρτηση που διαγράφει τα δεδομένα (γραμμή 132, Κώδικας 53). Με την διαγραφή των δεδομένων ο χρήστης ανακατευθύνεται στην αρχική σελίδα του aMazeD (γραμμή 133).

Κώδικας 53: Διαγραφή δεδομένων χρήστη στο αρχείο result.js

```
131  document.getElementById("clear").addEventListener("click", function() {
132      window.localStorage.clear();
133      window.location = "maze.html";
134  });
```

5.8. Αξιολόγηση του εργαλείου aMazeD

5.8.1. Εισαγωγή

Η υλοποίηση της αξιολόγησης του εργαλείου aMazeD δεν είναι στους στόχους της παρούσας διπλωματικής καθώς δεν ήταν δυνατή η πρόσβαση σε μαθητές με ηλικιακό εύρος 12-18 . Ωστόσο το παιχνίδι αξιολογήθηκε στα πλαίσια της έρευνας της κ. Χριστίνας Τίκβα με αντικείμενο το κατά πόσο οι μαθητές αντιλαμβάνονται την αποτελεσματικότητα ενός εργαλείου ανάπτυξης ΥΣ, που χρησιμοποιεί την Μάθηση με Υποστήριξη, στην ανάπτυξη της ΥΣ. Η κ. Τίκβα ενέκρινε την χρήση των αποτελεσμάτων της έρευνάς της στην παρούσα εργασία.

5.8.2. Περιγραφή της έρευνας

Τα ερωτήματα στα οποία προσανατολίζεται η έρευνα είναι τα εξής

- Αντιλαμβάνονται οι μαθητές το εργαλείο ανάπτυξης ΥΣ που κάνει χρήση της Μάθησης με Υποστήριξη ως εύκολο στην χρήση του;
- Αντιλαμβάνονται οι μαθητές το εργαλείο ανάπτυξης ΥΣ που κάνει χρήση της Μάθησης με Υποστήριξη ως αποτελεσματικό στην ανάπτυξη της ΥΣ;
- Αντιλαμβάνονται οι μαθητές τα εργαλεία υποστήριξης του εργαλείου ανάπτυξης ΥΣ που κάνει χρήση της Μάθησης με Υποστήριξη ως αποτελεσματικά για την ανάπτυξη της ΥΣ;

Το δείγμα της έρευνας αποτελείται από 28 μαθητές ενός Γυμνασίου, πιο συγκεκριμένα το 36% του δείγματος ήταν μαθητές της Α Γυμνασίου, το 39% ήταν μαθητές της Β Γυμνασίου και το 25% ήταν μαθητές της Γ Γυμνασίου. Το 46% του δείγματος είναι αγόρια και το 75% του δείγματος δήλωσε ότι έχει προηγούμενη εμπειρία με τον προγραμματισμό.

Η έρευνα έλαβε χώρα κατά την διάρκεια 2 διδακτικών ωρών[18] στο επίσημο ωρολόγιο πρόγραμμα του σχολείου. Οι μαθητές κλήθηκαν να παίξουν το παιχνίδι aMazeD για μια ώρα και έπειτα να συμπληρώσουν ένα ερωτηματολόγιο για 30 λεπτά.

Το ερωτηματολόγιο είναι σχεδιασμένο σύμφωνα με το μοντέλο TAM[19] και χωρίζεται σε τέσσερις ενότητες που αποτελούν τις διαστάσεις της κλίμακας

1. Αντιλαμβανόμενη Ευκολία Χρήσης
2. Αντιλαμβανόμενη Χρησιμότητα

[18] 2 διδακτικές ώρες αντιστοιχούν σε 1 ώρα και 30 λεπτά

[19] Technology Acceptance Model

3. Συμπεριφορά - Στάση
4. Προσβασιμότητα

Οι ερωτήσεις είναι δοσμένες σε κλίμακα Likert με εύρος τιμών από 1 ως 5, με την τιμή 1 να αντιστοιχεί στο «Διαφωνώ έντονα» και την τιμή 5 στο «Συμφωνώ έντονα». Οι τέσσερις ενότητες ελέγχθηκαν ως προς την αξιοπιστία εσωτερικής συνέπειας με το α του Cronbach να είναι ίσο με α=0,761>0,7, επομένως η κλίμακα θεωρείται αξιόπιστη.
Εκτός από τις 4 ενότητες και τα δημογραφικά στοιχεία οι μαθητές κλήθηκαν να απαντήσουν την ανοιχτού τύπου ερώτηση «Γράψτε μερικά λόγια για την εμπειρία σας με το παιχνίδι aMazeD. Τι σας άρεσε και τι δεν σας άρεσε; Τι σας εντυπωσίασε;»

5.8.3. Αποτελέσματα της έρευνας

Όσον αφορά την Αντιλαμβανόμενη Ευκολία Χρήσης το 64,3% του δείγματος βρήκε το εργαλείο aMazeD εύκολο/πολύ εύκολο στην χρήση και το 3.6% διαφώνησε ότι το παιχνίδι είναι εύκολο στην χρήση, ενώ το ποσοστό του δείγματος που θεωρεί εύκολο το να μάθει να χρησιμοποιεί ένα εργαλείο προγραμματισμού και ΥΣ είναι 60,7% .
Για την Αντιλαμβανόμενη Χρησιμότητα το 92,9% του δείγματος συμφώνησε ότι το aMazeD μπορεί να βελτιώσει την κατανόησή τους ως προς τις πρακτικές της ΥΣ, το 57,1% του δείγματος συμφώνησε ότι το παιχνίδι θα τους βοηθούσε να μάθουν πιο εύκολα έννοιες και πρακτικές της ΥΣ, ενώ το 7,2% διαφώνησε ή διαφώνησε έντονα στο τελευταίο. Η ανατροφοδότηση και η σχετική βοήθεια των επιπέδων κρίθηκε από το 75% του δείγματος αρκετή για την επίλυση των επιπέδων, από το 67,9% χρήσιμη και από το 53,6% βοηθητική για την κατανόηση των εννοιών και των πρακτικών της ΥΣ. Ενώ με την τελευταία δήλωση διαφώνησε ή διαφώνησε έντονα το 17,1% του δείγματος
Στην ενότητα Συμπεριφορά-Στάση το 82,1% του δείγματος έχει θετική στάση ως προς την μάθηση ΥΣ μέσω παιχνιδιών όπως το aMazeD ενώ παράλληλα το 92,9% του δείγματος δήλωσε ότι έχει θετική στάση ως προς την μάθηση ΥΣ μέσω παιχνιδιών γενικά. Ενώ το 3,6% έχει αρνητική στάση και στις δυο δηλώσεις.
Τέλος, στην ενότητα Προσβασιμότητα, το 60,7% συμφώνησε ή συμφώνησε έντονα ότι δεν είχε δυσκολία στην χρήση του παιχνιδιού aMazeD, ενώ το 14,3% αντιμετώπισε δυσκολίες κατά την χρήση του.
Στην τελευταία ερώτηση ανοιχτού τύπου καταγράφηκαν 25 απαντήσεις και κωδικοποιήθηκαν από την κ. Τίκβα σε δυο κατηγορίες την συνολική εικόνα του παιχνιδιού και την εμπειρία από την χρήση του παιχνιδιού aMazeD σε συνάρτηση με την ΥΣ και τον

προγραμματισμό. Έντεκα μαθητές απάντησαν ότι το παιχνίδι ήταν «ωραίο», «πολύ ωραίο», «ενδιαφέρον» και «απαιτητικό». Τρεις μαθητές εστίασαν τα σχόλια τους στην ευκολία χρήσης του παιχνιδιού, δυο μαθητές εξέφρασαν μέτρια ή αρνητικά σχόλια για το παιχνίδι, ενώ τρεις μαθητές εξέφρασαν θετικά σχόλια για την βοήθεια που παρέχει το παιχνίδι. Τέλος σύμφωνα με την κ. Τίκβα από τα σχόλια των μαθητών μπορεί να εξαχθεί το συμπέρασμα ότι στην πλειοψηφία τους βρήκαν το παιχνίδι aMazeD αποτελεσματικό για την εκμάθηση ΥΣ και προγραμματισμού.

5.8.4. Περίληψη

Στο κεφάλαιο αυτό παρουσιάστηκε η έρευνα της κ. Τίκβα για το κατά πόσο οι μαθητές αντιλαμβάνονται το εργαλείο ανάπτυξης ΥΣ aMazeD και τα χαρακτηριστικά του ως εύκολα στη χρήση και αποτελεσματικά ως προς την εκμάθηση ΥΣ. Η έρευνα δεν πραγματοποιήθηκε στα πλαίσια της διπλωματικής εργασίας αλλά χρησιμοποιείται με την συγκατάθεση της κ. Τίκβα για την συμπερασματική αξιολόγηση του παιχνιδιού aMazeD.

5.9. Περίληψη

Σε αυτό το κεφάλαιο παρουσιάσαμε τον κώδικα του εργαλείου aMazeD και τον τρόπο με τον οποίο σχεδιάστηκε το παιχνίδι για να ικανοποιεί τις λειτουργικότητες που είδαμε στο Κεφάλαιο 4. Αρχικά παρουσιάστηκε η δομή των αρχείων του παιχνιδιού και ο τρόπος με τον οποίο συνδέονται μεταξύ τους. Έπειτα παρουσιάστηκε και αναλύθηκε ο κώδικας που γράφτηκε ή τροποποιήθηκε για την διεπαφή του παιχνιδιού, την προσθήκη χρονομέτρου, τα πλήκτρα του παιχνιδιού, την ανάδραση και την παροχή βοήθειας στον χρήστη και την αποθήκευση των δεδομένων του χρήστη τοπικά. Τέλος παραθέτουμε μια συνοπτική παρουσίαση της αξιολόγησης του παιχνιδιού, η έρευνα έγινε ανεξάρτητα από την παρούσα διπλωματική εργασία, ωστόσο κρίθηκε σκόπιμη η συμπερίληψή της για την πλήρη παρουσίαση και ανάλυση του εκπαιδευτικού εργαλείου aMazeD.

6. Συμπεράσματα

Η ανάπτυξη δεξιοτήτων Υπολογιστικής Σκέψης μέσω της εκμάθησης προγραμματισμού είναι από τα πιο σύγχρονα προβλήματα της εκπαίδευσης. Ικανότητες όπως η επίλυση προβλημάτων, η αλγοριθμική σκέψη, η αξιολόγηση μια λύσης ως προς την αποδοτικότητά της, η εύρεση και αντιμετώπιση λαθών και άλλες ικανότητες που συνδέονται με την Υπολογιστική Σκέψη αποτελούν βασικό πυρήνα της εκπαίδευσης ίσης αξίας με τις αριθμητικές και γλωσσικές ικανότητες που πρέπει να καλλιεργούνται στους μαθητές. Σε αυτόν τον προσανατολισμό έχουν αναπτυχθεί πολλά περιβάλλοντα και εκπαιδευτικά παιχνίδια που στην πλειοψηφία τους χρησιμοποιούν εργαλεία οπτικού προγραμματισμού όπως είναι η βιβλιοθήκη της Blockly και η γλώσσα προγραμματισμού Scratch. Τα εκπαιδευτικά εργαλεία που μελετήθηκαν σε αυτή την εργασία είναι είτε παιχνίδια πρόκλησης είτε περιβάλλοντα οπτικού προγραμματισμού για την δημιουργία παιχνιδιών και πολυμέσων. Τα χαρακτηριστικά τους είναι ότι χρησιμοποιούν μπλοκ ή εικονίδια για την δημιουργία του προγράμματος από τον χρήστη, χρησιμοποιούν υπολογιστικές έννοιες όπως βρόγχους επανάληψης, συνθήκες, γεγονότα και υπολογιστικές πρακτικές όπως η εύρεση λαθών και η χρήση μεθόδων και συναρτήσεων, ενώ πολλά από αυτά επιτρέπουν την δημοσίευση των προγραμμάτων στο διαδίκτυο και την χρήση προγραμμάτων από άλλες πηγές για την ανάπτυξη πιο συνδυαστικών εφαρμογών.

Από την έρευνα που έγινε διαπιστώθηκε ότι πολλά από αυτά τα εργαλεία χρησιμοποιήθηκαν στη σχεδίαση εκπαιδευτικών προγραμμάτων για την ανάπτυξη της ΥΣ και σε έρευνες για την εφαρμογή πλαισίων ανάπτυξης ΥΣ και ένα από αυτά σχεδιάστηκε και αναπτύχθηκε εξ αρχής με βάση ενός πλαισίου ανάπτυξης ΥΣ. Στόχος αυτής της εργασίας ήταν η σχεδίαση και υλοποίηση ενός τέτοιου εκπαιδευτικού εργαλείου που θα κάνει χρήση εκπαιδευτικών μεθόδων και της Μάθησης με Υποστήριξη για την ανάπτυξη και αξιολόγηση της Υπολογιστικής Σκέψης σε μικρές ηλικίες. Το εργαλείο aMazeD που αναπτύχθηκε συνδυάζει τα χαρακτηριστικά των εκπαιδευτικών εργαλείων που μελετήθηκαν και βασίζεται στα Blockly Games. Για να τονιστεί η σημαντικότητα της μάθησης με Υποστήριξη το εργαλείο αναπτύχθηκε σε δυο εκδόσεις, η μια εκ των οποίων δεν την παρέχει.

Η πρώτη έκδοση του παιχνιδιού σχεδιάστηκε ώστε να ενσωματώνει την Μάθηση με Υποστήριξη με την παροχή ημιτελούς λύσης και μηνυμάτων βοήθειας που αφορούν την επεξήγηση των υπολογιστικών εννοιών που χρησιμοποιούνται και της λογικής του

παιχνιδιού. Επιπλέον τα προβλήματα ανά επίπεδο σχεδιάστηκαν για να χρησιμοποιούν υπολογιστικές έννοιες και πρακτικές που συναντάμε στο πλαίσιο ΥΣ των Brennan και Resnick. Σε κάθε επίπεδο του παιχνιδιού οι χρήστες αντιμετωπίζουν διαφορετικά προβλήματα και δέχονται διαφορετικού είδους ανάδραση κατάλληλα σχεδιασμένη για να τους οδηγεί στην λύση με στόχο όχι τη λύση του επιπέδου αυτή καθαυτή αλλά την επίλυση του επιπέδου με ένα αποδοτικό πρόγραμμα του οποίου την υλοποίηση μπορούν να δουν άμεσα με την χρήση του πλήκτρου «Παίξε». Ακόμα μια καινοτομία του παιχνιδιού είναι ότι παρέχει στον χρήστη τη δυνατότητα να αποθηκεύσει την λύση του ακόμα και αν δεν είναι σωστή, δίνοντάς του τη δυνατότητα να μεταβεί στο επόμενο επίπεδο και να ολοκληρώσει το παιχνίδι ακόμα και με επίπεδα που δεν έχουν σωστή λύση, αυτή την πρακτική δεν την συναντήσαμε σε κανένα από τα εκπαιδευτικά παιχνίδια που μελετήθηκαν, στα οποία η μετάβαση στο νέο επίπεδο γίνεται αποκλειστικά με την σωστή επίλυση του τρέχοντος επιπέδου.

Το μεγαλύτερο κομμάτι της εργασίας αφορά την σχεδίαση και υλοποίηση του παιχνιδιού και κατά την διάρκεια εκπόνησης της διπλωματικής αφιερώθηκε μεγάλο χρονικό διάστημα στην εξοικείωση με όλες τις νέες τεχνολογίες που χρησιμοποιήθηκαν. Το aMazeD στο μεγαλύτερο μέρος του αποτελείται από κώδικα JavaScript, χρησιμοποιεί εξαρτήσεις μεταξύ των αρχείων που υλοποιούνται με την χρήση των προτύπων Closure και η φυσική των εξαρτήσεων περιγράφεται με την χρήση της Python. Οι τροποποιήσεις γίνανε κυρίως στα αρχεία maze.js, maze.soy, turtle.js, turtle.soy και BlocklyGames.soy που υπήρχαν ήδη στα Blockly Games και αφορούν την προσθήκη συναρτήσεων και την τροποποίηση κώδικα ώστε να υλοποιηθούν οι προδιαγραφές που τέθηκαν κατά την σχεδίαση του παιχνιδιού. Επιπλέον στο παιχνίδι δημιουργήθηκαν δυο σελίδες οι Πληροφορίες και Αποτελέσματα που κάνουν χρήση του πλαισίου Bootstrap 4.3.1 .

Η αξιολόγηση του παιχνιδιού από την ανεξάρτητη έρευνα της κ. Τίκβα σε ένα δείγμα 28 μαθητών Γυμνασίου έδειξε ότι το aMazeD είναι ένα ικανοποιητικό εργαλείο για την εκμάθηση ΥΣ και προγραμματισμού.

6.1. Περιορισμοί της έρευνας

Η έρευνα στο θεωρητικό της κομμάτι περιορίστηκε μόνο σε περιβάλλοντα ανάπτυξης υπολογιστικής σκέψης και σε εργαλεία που είτε χρησιμοποιήθηκαν σε μελέτες για την ανάπτυξη ενός πλαισίου ΥΣ ή για την οργάνωση ενός διδακτικού μοντέλου που θα κάνει την χρήση του εργαλείου. Για την ανάπτυξη του παιχνιδιού χρειαζόμασταν ένα ανοιχτού

τύπου κώδικα παιχνίδι ως βάση που να χρησιμοποιεί εργαλεία οπτικού προγραμματισμού, για το λόγο αυτό επιλέχθηκαν τα Blockly Games. Ωστόσο το μέγεθος του κώδικα των παιχνιδιών είναι αρκετά μεγάλο και για την κατανόηση και τροποποίηση του αφιερώθηκε μεγάλο μέρος της ανάπτυξης του εργαλείου, επιπλέον δεν βρέθηκαν παρόμοια εργαλεία που να τροποποιούν τον κώδικα Blockly αν και μελετήθηκαν αρκετά που ήταν επηρεασμένα από τον Λαβύρινθο χωρίς ωστόσο ο κώδικας τους να μπορεί να αξιοποιηθεί.

Ο κώδικας των Blockly Games είναι μοιρασμένος σε πάρα πολλά αρχεία, ως αποτέλεσμα το παιχνίδι aMazeD έχει κρατήσει την πολυπλοκότητα όλων αυτώ των αρχείων και οι συναρτήσεις που προστέθηκαν επαναλαμβάνουν πολλές φορές το σώμα τους σε δυο αρχεία (maze.js-turtle.js και maze.soy-turtle.soy). Ένας ακόμα περιορισμός της έρευνας ήταν η περιορισμένη πληροφόρηση για το πως μπορεί να γίνει λήψη των παιχνιδιών σε περιβάλλον Windows καθώς οι σχεδιαστές των Blockly Games είχαν πρόθεση την τροποποίηση των παιχνιδιών με λειτουργικά τύπου Mac OS ή Linux. Ο τελευταίος περιορισμός είχε ως αποτέλεσμα να αφιερωθεί ένα πάρα πολύ μεγάλο κομμάτι της ανάπτυξης στην δοκιμή διαφόρων μεθόδων και πρακτικών για να μπορέσουμε να κάνουμε τροποποίηση του κώδικα και να τρέξουμε τα παιχνίδια τοπικά σε Windows 10.

Ένας ακόμα περιορισμός είναι ότι δεν δημιουργήθηκε βάση δεδομένων για το παιχνίδι aMazeD. Αυτό οφείλεται αρχικά στην επιρροή πολλών εφαρμογών που μελετήθηκαν και κρατάνε τα αρχεία του χρήστη τοπικά ή και καθόλου καθώς απευθύνονται σε μικρές ηλικίες και θα μπορούσε να υπάρχει κόλλημα με τους νόμους για την Προστασία Προσωπικών Δεδομένων που διαφοροποιούνται ανά χώρα. Με την πρακτική αυτή περιορίζεται η αξιολόγηση της ανάπτυξης της ΥΣ στους χρήστες καθώς ο ερευνητής θα πρέπει να ζητήσει τα αποτελέσματα από κάθε μαθητή ξεχωριστά.

Τέλος η βιβλιοθήκη της Blockly αν και έχει αρκετά παραδείγματα για τον τρόπο που γίνεται η εισαγωγή της σε απλές εφαρμογές δεν παρέχει αρκετές οδηγίες για την ανάπτυξη με πρότυπα Closure και αρχεία Python ή για την τροποποίηση του διερμηνέα JS Interpreter, αυτό είχε ως αποτέλεσμα να αλλάξουμε τον αρχικό σχεδιασμό στα επίπεδα 7-10 καθώς τα προβλήματα που είχαμε σχεδιάσει δεν μπορούσαν να αποτυπωθούν στον κώδικα και να τρέξουν σωστά.

6.2. Προτάσεις

Ο κώδικας του παιχνιδιού aMazeD μπορεί να βελτιωθεί σε μεγάλο βαθμό αφαιρώντας τις περιττές συναρτήσεις και δημιουργώντας κοινές συναρτήσεις στα αρχεία του φακέλου

common. Επιπλέον το παιχνίδι aMazeD είναι μοναδικό ως προς τον τρόπο που διαχειρίζεται και τροποποιεί τον κώδικα των Blockly Games και η ανάπτυξή του βασίστηκε πολύ στην δομή του κώδικα τους ενώ θα μπορούσε να υιοθετεί μόνο τα στοιχεία που είναι απαραίτητα και να αναπτυχθεί ανεξάρτητα, με την χρήση κάποιας άλλης τεχνολογίας. Επιπλέον θα μπορούσε να γίνει μια μελέτη σε σχέση με τα δεδομένα των μαθητών που μπορούν να αποθηκευτούν σε μια βάση δεδομένων και η χρήση της για μια ποιοτικότερη αξιολόγηση της ανάπτυξης της ΥΣ μέσα από το aMazeD. Τέλος το παιχνίδι μπορεί να αξιολογηθεί περαιτέρω και να εξελιχθούν τα προβλήματα των επιπέδων και τα μηνύματα βοήθειας που χρησιμοποιούνται ώστε να περιλαμβάνουν περισσότερες έννοιες και πρακτικές της ΥΣ.

Κατάλογος αναφορών - παραπομπών

Κατάλογος αναφορών

Brennan, K., & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. *Annual American Educational Research Association Meeting*, (σσ. 1-25). Vancouver, BC, Canada. Ανάκτηση από: http://web.media.mit.edu/~kbrennan/files/Brennan_Resnick_AERA2012_CT.pdf

Chao, P.-Y. (2016). Exploring students' computational practice, design and performance of problem-solving through a visual programming environment. *Computers & Education*(95), σσ. 202-215. Ανάκτηση από https://doi.org/10.1016/j.compedu.2016.01.010

Eguiluz, A., Garaizar, P., & Guenaga, M. (2017). *An Evaluation of Open Digital Gaming Platforms for Developing Computational Thinking Skills.* Retrieved from https://www.intechopen.com/books/simulation-and-gaming/an-evaluation-of-open-digital-gaming-platforms-for-developing-computational-thinking-skills

Fraser, N. (2015). *Ten Things we' ve learned from Blockly.* Ανάκτηση από https://developers.google.com/blockly/publications/papers/TenThingsWeveLearnedFromBlockly.pdf

Giannakoulas , A., & Xinogalos , S. (2020). A Review of Educational Games for Teaching Programming to Primary School Students. Στο *Handbook of Research on Tools for Teaching Computational Thinking in P-12 Education* (σσ. 1-30). IGI Global. doi:10.4018/978-1-7998-4576-8.ch001

Giannakoulas, A., & Xinogalos, S. (2018, September). A pilot study on the effectiveness and acceptance of an educational game for teaching programming concepts to primary school students. *EDUCATION AND INFORMATION TECHNOLOGIES, 23*(5), σσ. 2029-2052. doi:10.1007/s10639-018-9702-x

Humphreys, S., Csizmadia, A., Curzon, P., Dorling, M., Ng, T., Selby, S., & Wooland, J. (2015). *Computational Thinking - A guide for teachers.* Computing At School.

ISTE, & CSTA. (2011). *Computational thinking leadership toolkit (First Edition).* Ανάκτηση από https://cdn.iste.org/www-root/2020-10/ISTE_CT_Leadership_Toolkit_booklet.pdf

João, P., Nuno, D., Sampaio , F. F., & Ana, P. (2019). A Cross-analysis of Block-based and Visual Programming Apps with Computer Science Student-Teachers. *Education Sciences, 9*(3), σ. 181. Ανάκτηση από https://doi.org/10.3390/educsci9030181

Kanelopoulou, I., Garaizar , P., & Guenaga , M. (2021). First Steps Towards Automatically Defining the Difficulty of Maze-Based Programming Challenges. *IEEE ACCESS, 9*, σσ. 64211-64223. doi:10.1109/ACCESS.2021.3075027

Karakasis, C., & Xinogalos, S. (2020). BlocklyScript: Design and Pilot Evaluationof an RPG Platform Game for Cultivating Computational Thinking Skills to Young Students. *Informatics in Education, 19*, σσ. 641-668. doi:10.15388/infedu.2020.28

Kazimoglu, C., Kiernan, M., Bacon, L., & MacKinnon, L. (2012). Learning Programming at the Computational Thinking Level via Digital Game-Play. *Procedia Computer Science*(9), σσ. 522-531. Ανάκτηση από www.sciencedirect.com

Lye, S., & Koh, J. (2014). Review on teaching and learning of computational thinking through programming: What is next for K-12? *Computers In Human Behavior, 41*, σσ. 51-61. doi:10.1016/j.chb.2014.09.012

Marron, A., Weiss, G., & Wiener , G. (2012). A decentralized approach for programming interactive applications with Javascript and Blockly. *Conference: Proceedings of the 2nd edition on Programming systems, languages and applications based on actors, agents, and decentralized control abstractions.* Ανάκτηση από: https://www.researchgate.net/publication/262246746_A_decentralized_approach_for_programming_interactive_applications_with_JavaScript_and_blockly

Mathrani, A., Christian, S., & Ponder-Sutton, A. (2016). PlayIT: Game Based Learning Approach for Teaching Programming Concepts. *EDUCATIONAL TECHNOLOGY & SOCIETY, 19*(2), σσ. 5-17.

Microsoft. (χ.χ.). *Kodu Game Lab.* Ανάκτηση από http://www.kodugamelab.com/

MIT. (χ.χ.). *An Introduction to Makefiles.* Ανάκτηση από web.mit.edu: https://web.mit.edu/gnu/doc/html/make_2.html

Moreno Leon, J., Roman Gonzalez , M., & Robles, G. (2015). Dr. Scratch: Automatic Analysis of Scratch Projects to Assess and Foster Computational Thinking. *RED: Revista de Educación a Distancia*(46). Ανάκτηση από https://www.researchgate.net/publication/281714025

Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas.* New York: Basic Books, Inc., Publishers.

Papert, S. (1996). An exploration in the space of mathematics educations. *International Journal of Computers for Mathematical Learning, 1*, σσ. 95-123.

Run Marco! (χ.χ.). Ανάκτηση από https://runmarco.allcancode.com/

Touretzky, D. S., Marghitu, D., Ludi, S., Bernstein, D., & Ni, L. (2013). Accelerating k-12 computational thinking using Scaffolding, Staging, and Abstraction. *SIGCSE, 44th ACM Technical Symposium on Computer Science Education* (σσ. 609-614). Association for Computing Machinery. doi:10.1145/2445196.2445374

TSIOTRAS, D., & XINOGALOS, S. (2021). Investigating the Perceived Player Experience and Short-term Learning of the Text-based Java Programming Serious Game "Rise of the Java Emperor". *Informatics in Education, 20*(1), σσ. 153–170. doi:10.15388/infedu.2021.08

Wing, J. M. (2006, March). Computational Thinking. *Communications of the ACM*, σσ. 33-35.

Wood, D., Bruner, J., & Ross, G. (1976). The role of tutoring in problem solving. *The Journal of Child Psychology and Psychiatry*, σσ. 89-100. doi:https://doi.org/10.1111/j.1469-7610.1976.tb00381.x

Yağcı, M. (2019). A valid and reliable tool for examining computational. *Education and Information Technologies*(24), σσ. 929-951.

Παράρτημα Α

Εγκατάσταση Blockly Games σε περιβάλλον Windows 10

Εγκατάσταση υποσύστημα Linux

Για την εγκατάσταση των Blockly Games είναι απαραίτητη η εντολή make που υπάρχει μόνο σε περιβάλλον Linux. Αν κάποιος είναι χρήστης Windows 10 μπορεί να παρακάμψει αυτό το πρόβλημα με την εγκατάσταση του δωρεάν λογισμικού Ubuntu από το Microsoft Store. Το Ubuntu είναι ένα τερματικό γραμμής εντολών. Απαραίτητη προϋπόθεση είναι το λειτουργικό του σύστημα να έχει κάνει την αναβάθμιση Windows 10 Fall Creators (Οκτώβρης 2017) . Στην αναβάθμιση αυτή περιέχεται το υποσύστημα Windows Subsystem for Linux (wsl) που είναι απαραίτητο για την λειτουργία του τερματικού Ubuntu.

Τα βήματα που πρέπει να πραγματοποιηθούν είναι τα εξής:

- Άνοιγμα του PowerShell με δικαιώματα διαχειριστή και ενεργοποίηση του wsl με την εντολή
  `dism.exe /online /enable-feature /featurename:Microsoft-Windows-Subsystem-Linux /all /norestart`
- Επανεκκίνηση των Windows.
- Εγκατάσταση της εφαρμογής Ubuntu από το Microsoft Store.
- Εκκίνηση της εφαρμογής από την έναρξη και ολοκλήρωση της διαδικασίας εγκατάστασης.
- Η εφαρμογή θα ζητήσει ένα όνομα χρήστη και έναν κωδικό πρόσβασης που θα χρειάζεστε κάθε φορά για την εγκατάσταση νέων πακέτων.

Αφού ολοκληρωθεί η εγκατάσταση χρησιμοποιώντας την εντολή wsl (σε περιβάλλον PowerShell) σε οποιοδήποτε κατάλογο (directory) μπορούμε να κάνουμε χρήση του τερματικού γραμμής εντολών του Ubuntu σε αυτό τον κατάλογο.

Λήψη και ανάπτυξη (build) των Blockly Games

Ο κώδικας των Blockly Games είναι διαθέσιμος στο GitHub, από εκεί μπορούμε να κάνουμε λήψη του συμπιεσμένου αρχείου που περιέχει όλα τα αρχεία και τους

φακέλους για να τρέξουμε τα παιχνίδια τοπικά στον υπολογιστή μας και να κάνουμε αλλαγές στα template.soy και στα αρχεία js.

Αφού αποσυμπιέσουμε το αρχείο σε έναν φάκελο, ακολουθούμε τα εξής βήματα

- Από την γραμμή εντολών (PowerShell ή Windows Terminal) με τη χρήση της εντολής cd (change directory) μετακινούμαστε στο κατάλογο που έχουμε αποθηκεύσει αποσυμπιεσμένο αρχείο.
- Δίνουμε την εντολή wsl.
- Δίνουμε την εντολή sudo apt install make , είναι απαραίτητη μόνο την πρώτη φορά που θα δουλέψουμε στο περιβάλλον του Ubuntu.
- Δίνουμε την εντολή made deps για να αναπτύξουμε την εφαρμογή. Ακολουθούμε τις οδηγίες για την εγκατάσταση των απαραίτητων πακέτων.
- Μπορούμε να αναπτύξουμε όλα τα παιχνίδια στα Αγγλικά χρησιμοποιώντας την εντολή `make en` ή `make languages` για όλες τις γλώσσες.
  ή κάθε παιχνίδι ξεχωριστά χρησιμοποιώντας τις εντολές

```
make index-en
make puzzle-en
make maze-en
make bird-en
make turtle-en
make movie-en
make music-en
make pond-docs-en
make pond-tutor-en
make pond-duck-en
make gallery-en
```

- Τέλος ένα παιχνίδι μπορούμε να το αναπτύξουμε σε όλες τις γλώσσες χρησιμοποιώντας την εντολή
  `make όνομα_παιχνιδιού`

Παράρτημα Β

Βασικές συναρτήσεις και σελίδες του παιχνιδιού aMazeD.

Β1. Η συνάρτηση Maze.animate

```
1495  Maze.animate = function() {
1496      var action = Maze.log.shift();
1497      if (!action) {
1498          BlocklyInterface.highlight(null);
1499          return;
1500      }
1501      BlocklyInterface.highlight(action[1]);
1502
1503      switch (action[0]) {
1504          case 'north':
1505              Maze.schedule([Maze.pegmanX, Maze.pegmanY, Maze.pegmanD * 4],
                  [Maze.pegmanX, Maze.pegmanY - 1, Maze.pegmanD * 4]);
1506              Maze.pegmanY--;
1507              break;
1508          case 'east':
1509              Maze.schedule([Maze.pegmanX, Maze.pegmanY, Maze.pegmanD * 4],
                  [Maze.pegmanX + 1, Maze.pegmanY, Maze.pegmanD * 4]);
1510              Maze.pegmanX++;
1511              break;
1512          case 'south':
1513              Maze.schedule([Maze.pegmanX, Maze.pegmanY, Maze.pegmanD * 4],
                  [Maze.pegmanX, Maze.pegmanY + 1, Maze.pegmanD * 4]);
1514              Maze.pegmanY++;
1515              break;
1516          case 'west':
1517              Maze.schedule([Maze.pegmanX, Maze.pegmanY, Maze.pegmanD * 4],
                  [Maze.pegmanX - 1, Maze.pegmanY, Maze.pegmanD * 4]);
1518              Maze.pegmanX--;
1519              break;
1520          case 'look_north':
1521              Maze.scheduleLook(Maze.DirectionType.NORTH);
1522              break;
1523          case 'look_east':
1524              Maze.scheduleLook(Maze.DirectionType.EAST);
1525              break;
1526          case 'look_south':
1527              Maze.scheduleLook(Maze.DirectionType.SOUTH);
1528              break;
1529          case 'look_west':
1530              Maze.scheduleLook(Maze.DirectionType.WEST);
1531              break;
1532          case 'fail_forward':
1533              Maze.scheduleFail(true);
1534              break;
```

```
1535            case 'fail_backward':
1536                Maze.scheduleFail(false);
1537                break;
1538            case 'left':
1539                Maze.schedule([Maze.pegmanX, Maze.pegmanY, Maze.pegmanD * 4],
                    [Maze.pegmanX, Maze.pegmanY, Maze.pegmanD * 4 - 4]);
1540                Maze.pegmanD = Maze.constrainDirection4(Maze.pegmanD - 1);
1541                break;
1542            case 'right':
1543                Maze.schedule([Maze.pegmanX, Maze.pegmanY, Maze.pegmanD * 4],
                    [Maze.pegmanX, Maze.pegmanY, Maze.pegmanD * 4 + 4]);
1544                Maze.pegmanD = Maze.constrainDirection4(Maze.pegmanD + 1);
1545                break;
1546                //βάλαμε την παρακάτω περίπτωση αν έχουμε fail και πατάμε submit
1547            case 'end':
1548                BlocklyInterface.saveToLocalStorage();
1549                setTimeout(BlocklyDialogs.endOfLevel, 1000);
1550                break;
1551            case 'finish':
1552                Maze.scheduleFinish(true);
1553                BlocklyInterface.saveToLocalStorage();
1554                setTimeout(BlocklyDialogs.congratulations, 1000);
1555        }
1556
1557        Maze.pidList.push(setTimeout(Maze.animate, Maze.stepSpeed * 5));
1558    };
```

Β2. Η συνάρτηση Turtle.isCorrect

```
122   Turtle.isCorrect = function(pixelErrors) {
123       console.log('Pixel errors: ' + pixelErrors);
124       if (pixelErrors > 100) {
125           // Too many errors.
126           return false;
127       }
128       var blockCount = BlocklyInterface.workspace.getAllBlocks().length;
129       if ((BlocklyGames.LEVEL == 8 && blockCount > 5) ||
130           (BlocklyGames.LEVEL == 9 && blockCount > 4)) {
131           var content = document.getElementById('helpUseLoop');
132           var style = {
133               'width': '30%',
134               'left': '35%',
135               'top': '12em'
136           };
137           BlocklyDialogs.showDialog(content, null, false, true, style,
138               BlocklyDialogs.stopDialogKeyDown);
139           BlocklyDialogs.startDialogKeyDown();
140           return false;
141       }
142       return true;
143   };
```

Β3. Η συνάρτηση Turtle.checkAnswer

```
1057  Turtle.checkAnswer = function() {
1058      // Compare the Alpha (opacity) byte of each pixel in the user's image and
1059      // the sample answer image.
1060      var userImage =
1061          Turtle.ctxScratch.getImageData(0, 0, Turtle.WIDTH, Turtle.HEIGHT);
1062      var answerImage =
1063          Turtle.ctxAnswer.getImageData(0, 0, Turtle.WIDTH, Turtle.HEIGHT);
1064      var len = Math.min(userImage.data.length, answerImage.data.length);
1065      var delta = 0;
1066      // Pixels are in RGBA format.  Only check the Alpha bytes.
1067      for (var i = 3; i < len; i += 4) {
1068          // Check the Alpha byte.
1069          if (Math.abs(userImage.data[i] - answerImage.data[i]) > 64) {
1070              delta++;
1071              console.log(delta);
1072          }
1073      }
1074
1075      if (Turtle.isCorrect(delta)) {
1076          //αποθήκευσε δεδομένα επιτυχίας
1077          document.getElementById("msgSuccess").style.display = 'inline';
1078          Turtle.levelData[BlocklyGames.LEVEL - 7].score = 10 * BlocklyGames.
              LEVEL;
1079          Turtle.levelData[BlocklyGames.LEVEL - 7].result = 'Success';
1080          BlocklyInterface.saveToLocalStorage();
1081          Turtle.saveScore();
1082          BlocklyInterface.workspace.getAudioManager().play('win', 0.5);
1083          BlocklyDialogs.congratulations();
1084      } else if (!Turtle.isCorrect(delta)) {
1085          //αν δεν είναι επιτυχημένη η προσπάθεια αλλά πατήσει submit
1086          //αποθήκευσε δεδομένα αποτυχίας
1087          document.getElementById("msgFail").style.display = 'inline';
1088          Turtle.levelData[BlocklyGames.LEVEL - 7].score = 0;
1089          Turtle.levelData[BlocklyGames.LEVEL - 7].result = 'Failure';
1090          BlocklyInterface.saveToLocalStorage();
1091          Turtle.saveScore();
1092          BlocklyDialogs.endOfLevel();
1093      }
1094  };
```

B4. Run Button

Λαβύρινθος

```
1201  Maze.runButtonClick = function(e) {
1202      // Prevent double-clicks or double-taps.
1203      if (BlocklyInterface.eventSpam(e)) {
1204          return;
1205      }
1206
1207      Maze.levelData[BlocklyGames.LEVEL - 1].countPlay += 1;
1208      document.querySelector(".outcome__value--play").textContent = Maze.
          levelData[BlocklyGames.LEVEL - 1].countPlay;
1209
1210
1211      console.log(Maze.levelData);
1212
1213      BlocklyDialogs.hideDialog(false);
1214
1215      var runButton = document.getElementById('runButton');
1216      var resetButton = document.getElementById('resetButton');
1217      // Ensure that Reset button is at least as wide as Run button.
1218      if (!resetButton.style.minWidth) {
1219          resetButton.style.minWidth = runButton.offsetWidth + 'px';
1220      }
1221      runButton.style.display = 'none'; //changed from none
1222      resetButton.style.display = 'inline';
1223      Maze.reset(false);
1224      Maze.play();
1225  };
```

Ζωγραφική

```
651   Turtle.runButtonClick = function(e) {
652       // Prevent double-clicks or double-taps.
653       if (BlocklyInterface.eventSpam(e)) {
654           return;
655       }
656
657       Turtle.levelData[BlocklyGames.LEVEL - 7].countPlay += 1;
658       document.querySelector(".outcome__value--play").textContent = Turtle.
          levelData[BlocklyGames.LEVEL - 7].countPlay;
659
660       var runButton = document.getElementById('runButton');
661       var resetButton = document.getElementById('resetButton');
662       var submitButton = document.getElementById('submitButton');
663       // Ensure that Reset button is at least as wide as Run button.
664       if (!resetButton.style.minWidth) {
665           resetButton.style.minWidth = runButton.offsetWidth + 'px';
666       }
667       runButton.style.display = 'none';
668       submitButton.style.display = 'inline';
669       resetButton.style.display = 'inline';
670       document.getElementById('spinner').style.visibility = 'visible';
671       Turtle.play();
672
673   };
```

B5. Reset Button

Λαβύρινθος

```
1152  Maze.reset = function(first) {
1153      // Kill all tasks.
1154      for (var i = 0; i < Maze.pidList.length; i++) {
1155          clearTimeout(Maze.pidList[i]);
1156      }
1157      Maze.pidList = [];
1158
1159      // Move Pegman into position.
1160      Maze.pegmanX = Maze.start_.x;
1161      Maze.pegmanY = Maze.start_.y;
1162
1163      if (first) {
1164          // Opening animation.
1165          Maze.pegmanD = Maze.startDirection + 1;
1166          Maze.scheduleFinish(false);
1167          Maze.pidList.push(setTimeout(function() {
1168              Maze.stepSpeed = 100;
1169              Maze.schedule([Maze.pegmanX, Maze.pegmanY, Maze.pegmanD * 4],
                  [Maze.pegmanX, Maze.pegmanY, Maze.pegmanD * 4 - 4]);
1170              Maze.pegmanD++;
1171          }, Maze.stepSpeed * 5));
1172      } else {
1173          Maze.pegmanD = Maze.startDirection;
1174          Maze.displayPegman(Maze.pegmanX, Maze.pegmanY, Maze.pegmanD * 4);
1175      }
1176
1177      // Move the finish icon into position.
1178      var finishIcon = document.getElementById('finish');
1179      finishIcon.setAttribute('x', Maze.SQUARE_SIZE * (Maze.finish_.x + 0.5) -
1180          finishIcon.getAttribute('width') / 2);
1181      finishIcon.setAttribute('y', Maze.SQUARE_SIZE * (Maze.finish_.y + 0.6) -
1182          finishIcon.getAttribute('height'));
1183
1184      // Make 'look' icon invisible and promote to top.
1185      var lookIcon = document.getElementById('look');
1186      lookIcon.style.display = 'none';
1187      lookIcon.parentNode.appendChild(lookIcon);
1188      var paths = lookIcon.getElementsByTagName('path');
1189      for (var i = 0, path;
1190          (path = paths[i]); i++) {
1191          path.setAttribute('stroke', Maze.SKIN.look);
1192      }
1193  };
```

Ζωγραφική

```
712  Turtle.resetButtonClick = function(e) {
713      // Prevent double-clicks or double-taps.
714      if (BlocklyInterface.eventSpam(e)) {
715          return;
716      }
717      var runButton = document.getElementById('runButton');
718      var submitButton = document.getElementById('submitButton');
719      runButton.style.display = 'inline';
720      submitButton.style.display = 'inline';
721      document.getElementById('resetButton').style.display = 'inline';
722      document.getElementById('spinner').style.visibility = 'hidden';
723      BlocklyInterface.workspace.highlightBlock(null);
724      Turtle.reset();
725      Turtle.showHelp();
726
727      // Image cleared; prevent user from submitting to gallery.
728      // Turtle.canSubmit = false;
729  };
```

B6. Submit Button

Ζωγραφική

```
679  Turtle.submitButtonClick = function(e) {
680      // Prevent double-clicks or double-taps.
681      if (BlocklyInterface.eventSpam(e)) {
682          return;
683      }
684
685      if (timer) {
686          clearInterval(timer);
687          Turtle.levelData[BlocklyGames.LEVEL - 7].time = document.
         getElementById('capacity').textContent;
688          // console.log(Turtle.levelData[BlocklyGames.LEVEL - 7]);
689      }
690
691      var runButton = document.getElementById('runButton');
692      var resetButton = document.getElementById('resetButton');
693      var submitButton = document.getElementById('submitButton');
694      // Ensure that Reset button is at least as wide as Run button.
695      if (!resetButton.style.minWidth) {
696          resetButton.style.minWidth = runButton.offsetWidth + 'px';
697      }
698      runButton.style.display = 'none';
699      submitButton.style.display = 'none';
700      resetButton.style.display = 'none';
701      document.getElementById('spinner').style.visibility = 'none';
702      Turtle.execute();
703  };
```

Λαβύρινθος

```
1230  Maze.submitButtonClick = function(e) {
1231      // Prevent double-clicks or double-taps.
1232      if (BlocklyInterface.eventSpam(e)) {
1233          return;
1234      }
1235      BlocklyDialogs.hideDialog(false);
1236
1237      if (timer) {
1238          clearInterval(timer);
1239          Maze.levelData[BlocklyGames.LEVEL - 1].time = document.getElementById
          ('capacity').textContent;
1240          console.log(Maze.levelData[BlocklyGames.LEVEL - 1]);
1241      }
1242
1243
1244      var runButton = document.getElementById('runButton');
1245      var resetButton = document.getElementById('resetButton');
1246      var submitButton = document.getElementById('submitButton');
1247      // Ensure that Reset button is at least as wide as Run button.
1248      if (!resetButton.style.minWidth) {
1249          resetButton.style.minWidth = runButton.offsetWidth + 'px';
1250      }
1251      runButton.style.display = 'none'; //changed from none
1252      resetButton.style.display = 'none';
1253      submitButton.style.display = 'none';
1254
1255      Maze.reset(false);
1256      Maze.execute();
1257
1258      // κώδικας για αποθήκευση του score σε κάθε level στο localStorage
1259      // Store
1260      var lname = "maze_level" + BlocklyGames.LEVEL;
1261      window.localStorage.setItem(lname, JSON.stringify(Maze.levelData
          [BlocklyGames.LEVEL - 1]));
1262
1263      document.querySelector(".outcome__value--score").textContent = Maze.
          levelData[BlocklyGames.LEVEL - 1].score;
1264      if (window.localStorage.getItem('total-score')) {
1265          totalScore = totalScore + Maze.levelData[BlocklyGames.LEVEL - 1].
              score;
1266          document.querySelector('.outcome__value--total__score').textContent
              = totalScore;
1267          window.localStorage.setItem('total-score', totalScore);
1268      }
1269  };
```

Β7. Η συνάρτηση BlocklyDialogs.endOfLevel

```
420  BlocklyDialogs.endOfLevel = function() {
421      var content = document.getElementById('dialogDone2');
422      var style = {
423          width: '40%',
424          left: '30%',
425          top: '3em'
426      };
427
428      // Add the user's code.
429      if (BlocklyInterface.workspace) {
430          var linesText = document.getElementById('dialogLinesText2');
431          linesText.textContent = '';
432          var code = BlocklyInterface.executedJsCode;
433          code = BlocklyInterface.stripCode(code);
434          var noComments = code.replace(/\/\/[^\n]*/g, ''); // Inline comments.
435          noComments = noComments.replace(/\/\*.*\*\//g, ''); /* Block
             comments. */
436          noComments = noComments.replace(/[ \t]+\n/g, '\n'); // Trailing
             spaces.
437          noComments = noComments.replace(/\n+/g, '\n'); // Blank lines.
438          noComments = noComments.trim();
439          var lineCount = noComments.split('\n').length;
440          var pre = document.getElementById('containerCode2');
441          pre.textContent = code;
442          //αποθήκευση του κώδικα
443          var codeKey = 'maze-code' + BlocklyGames.LEVEL;
444          var codeValue = pre.textContent;
445          window.localStorage.setItem(codeKey, codeValue);
446
447          if (typeof prettyPrintOne == 'function') {
448              code = pre.innerHTML;
449              code = prettyPrintOne(code, 'js');
450              pre.innerHTML = code;
451          }
452          if (lineCount == 1) {
453              var text = BlocklyGames.getMsg('Games_linesOfCode1');
454          } else {
455              var text = BlocklyGames.getMsg('Games_linesOfCode2')
456                  .replace('%1', String(lineCount));
457          }
458          linesText.appendChild(document.createTextNode(text));
459      }
460
461      if (BlocklyGames.LEVEL < BlocklyGames.MAX_LEVEL) {
462          var text = BlocklyGames.getMsg('Games_nextLevel')
463              .replace('%1', String(BlocklyGames.LEVEL + 1));
464      } else {
465          var text = BlocklyGames.getMsg('Games_finalLevel');
466      }
467
468      var ok = document.getElementById('doneOk2');
469      ok.addEventListener('click', BlocklyInterface.nextLevel, true);
470      ok.addEventListener('touchend', BlocklyInterface.nextLevel, true);
```

```
471
472     BlocklyDialogs.showDialog(content, null, false, true, style,
473         function() {
474             document.body.removeEventListener('keydown',
475                 BlocklyDialogs.congratulationsKeyDown, true);
476         });
477     document.body.addEventListener('keydown',
478         BlocklyDialogs.congratulationsKeyDown, true);
479
480     document.getElementById('dialogDoneText2').textContent = text;
481 };
```

Β8. Κώδικας σελίδας about-el.html

```
<!DOCTYPE html>
<html lang="el">

<head>
    <meta charset="utf-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <!-- Bootstrap -->
    <link href="css/bootstrap-4.3.1.css" rel="stylesheet" />
    <style>
        body {
            background-color: #f3f3f3;
        }

        .info {
            padding: 20px;
            background-color: #4981fa;
            color: white;
        }

        .closebtn {
            margin-left: 15px;
            color: white;
            font-weight: bold;
            float: right;
            font-size: 22px;
            line-height: 20px;
            cursor: pointer;
            transition: 0.3s;
        }

        .closebtn:hover {
            color: black;
        }
    </style>
    <title>Πληροφορίες</title>
</head>

<body>
    <!-- navbar -->
    <nav class="navbar fixed-top navbar-expand-lg navbar-light bg-light">
        <a class="navbar-brand" href="maze.html">
            <img src="images/logo_amazed.png" width="120" alt="mazeGame">
        </a>

        <button class="navbar-toggler" type="button" data-toggle="collapse"
        data-target="#navbarSupportedContent1" aria-controls="navbarSupportedContent1"
        aria-expanded="false" aria-label="Toggle navigation"> <span class="navbar-toggler-icon"></
        span> </button>
        <div class="collapse navbar-collapse" id="navbarSupportedContent1">
            <ul class="navbar-nav mr-auto">
                <li class="nav-item"><a class="nav-link" href="maze.html">Επιστροφή στο Παιχνίδι</a></
                li>
                <li class="nav-item "><a class="nav-link" href="#info">Πληροφορίες</a></li>
                <li class="nav-item"><a class="nav-link" href="#functionality">Λειτουργικότητα</a></
                li>
                <li class="nav-item"><a class="nav-link" href="#blocks">Blocks</a></li>
                <li class="nav-item"><a class="nav-link" href="#results">Αποτελέσματα</a></li>
            </ul>
        </div>
    </nav>
    <header>
    </header>
```

```
58
59      <div class="container">
60          <!-- main -->
61          <main>
62
63              <section id="info" style="padding-top: 80px">
64                  <div class="row">
65                      <div class="col-12">
66                          <h3>Πληροφορίες</h3>
67                          <hr />
68                          <p> Το aMazeD είναι ένα εκπαιδευτικό παιχνίδι που δημιουργήθηκε με τη χρήση
                            της βιβλιοθήκης <a href='https://developers.google.com/blockly'
                            target="_blank">Blockly</a> και την επέκταση της λειτουργικότητας των
                            παιχνιδιών <a href="https://blockly.games/"
69                                  target="_blank">Blockly Games.</a></p>
70                          <p>Ο κώδικας του παιχνιδιού είναι διαθέσιμος στο <a href="https://github.com/
                            mmousiou/aMazeD" target="_blank">GitHub</a> .</p>
71                      </div>
72                  </div>
73              </section>
74
75
76              <section id="functionality" style="padding-top: 80px">
77                  <h4>Λειτουργικότητα παιχνιδιού</h4>
78                  <hr />
79                  <div class="container">
80                      <div class="row">
81                          <div class="col-12">
82                              <p>Το παιχνίδι αποτελείται από 10 επίπεδα τα οποία χωρίζονται σε δυο
                                κατηγορίες.</p>
83                              <ul>
84                                  <li>Τα επίπεδα 1-6 συνθέτουν την πρώτη κατηγορία του παιχνιδιού που
                                    είναι ένας Λαβύρινθος. Ο παίκτης με την κατάλληλη επιλογή block
                                    πρέπει να δημιουργήσει ένα πρόγραμμα για να φτάσει ο χαρακτήρας στο
                                    τέλος του λαβυρίνθου.</li>
85                                  <li> Τα επίπεδα 7-10 συνθέτουν την δεύτερη κατηγορία του παιχνιδιού
                                    που είναι η Ζωγραφική. Ο παίκτης πρέπει με την κατάλληλη επιλογή
                                    block να δημιουργήσει ένα πρόγραμμα που θα σχεδιάσει το ζητούμενο
                                    σχήμα.</li>
86                              </ul>
87                              <p> Για τον έλεγχο και του κώδικα και την υποβολή απάντησης ο χρήστης
                                έχει στην διάθεσή του τα ακόλουθα πλήκτρα:</p>
88                          </div>
89                      </div>

90                      <div class="row">
91                          <div class="col-lg-4 col-md-6 col-sm-12 text-center">
92                              <img class="rounded" alt="play" src="images/play.png"
                                data-holder-rendered="true">
93                              <h5>Πλήκτρο "Παίξε"</h5>
94                              <p>Πατώντας το "Παίξε" μπορείτε να δείτε την την κίνηση του παίκτη
                                ανάλογα με τον κώδικά σας. Όταν εκτελεστούν οι εντολές από τα blocks που
                                έχετε στοιβάξει θα πρέπει να πατήσετε επαναφορά για να επιστρέψει ο
                                παίκτης σας στην εκκίνηση.</p>
95                          </div>
96                          <div class="col-lg-4 col-md-6 col-sm-12 text-center">
97                              <img class="rounded" alt="reset" src="images/reset.png"
                                data-holder-rendered="true">
98                              <h5>Πλήκτρο "Επαναφορά"</h5>
99                              <p>Πατώντας "Επαναφορά" δεν εκτελείται ο κώδικας των blocks και
                                επιστρέφει ο παίκτης σας στην εκκίνηση. </p>
100                         </div>
101                         <div class="col-lg-4 col-md-6 col-sm-12 text-center">
102                             <img class="rounded" alt="submit" src="images/submit.png"
                                data-holder-rendered="true">
103                             <h5>Πλήκτρο "Υποβολή"</h5>
104                             <p>Με την "Υποβολή" υποβάλλετε την απάντησή σας είτε είναι σωστή είτε
                                λανθασμένη. Πρέπει να πατήσετε "Υποβολή" για να περάσετε στο επόμενο
                                επίπεδο. </p>
105                         </div>
106                     </div>
107                 </div>
108             </section>
```

```
110             <section id="blocks" style="padding-top: 80px">
111                 <h4>Blocks</h4>
112                 <hr />
113                 <div class="container mt-12 col-12">
114                     <div class="row justify-content-lg-start">
115                         <ul class="list-unstyled" style="padding-top: 10px">
116                             <li class="media">
117                                 <img class="mr-2" src="images/move_forward.png" alt="Generic
                                    placeholder image" />
118                                 <div class="media-body">
119                                     <h5 class="mt-0 mb-0">Προχώρησε ευθεία</h5>
120                                     Ο παίκτης προχωρά ευθεία για ένα βήμα.
121                                 </div>
122                             </li>
123                         </ul>
124                         <ul class="list-unstyled" style="padding-top: 10px">
125                             <li class="media">
126                                 <img class="mr-2" src="images/turn_left.png" alt="Generic placeholder
                                    image" />
127                                 <div class="media-body">
128                                     <h5 class="mt-0 mb-0">Στρίψε αριστερά/δεξιά</h5>
129                                     Ο παίκτης στρίβει επί τόπου αριστερά ή δεξιά ανάλογα με την
                                        επιλογή του χρήστη.
130                                 </div>
131                             </li>
132                         </ul>
133 >                       <ul class="list-unstyled" style="padding-top: 10px">···
142                         </ul>
143 >                       <ul class="list-unstyled" style="padding-top: 10px">···
151                         </ul>
152 >                       <ul class="list-unstyled" style="padding-top: 10px">···
160                         </ul>
161 >                       <ul class="list-unstyled" style="padding-top: 10px">···
169                         </ul>
170 >                       <ul class="list-unstyled" style="padding-top: 10px">···
178                         </ul>
179 >                       <ul class="list-unstyled" style="padding-top: 10px">···
187                         </ul>
188                         <ul class="list-unstyled" style="padding-top: 10px">
189 >                           <li class="media">···
195                             </li>
196                         </ul>
197                     </div>
198                 </div>
199             </section>
```

```
201             <section id="results" style="padding-top: 80px">
202                 <div class="row">
203                     <div class="col-12">
204                         <h3>Αποτελέσματα</h3>
205                         <hr />
206                         <p>Με την ολοκλήρωση του παιχνιδιού εμφανίζεται η σελίδα των αποτελεσμάτων με
                            τα δεδομένα ανά επίπεδο και την συνολική βαθμολογία του παιχνιδιού. Για να
                            εμφανιστούν τα αποτελέσματα είναι απαραίτητο να εισάγετε ένα όνομα και μια
                            έγκυρη
207                             διεύθυνση e-mail. </p>
208                         <div class="info">
209                             <span class='closebtn' onclick="this.parentElement.style.display='none';
                                ">&times;</span> Γίνεται μόνο στατιστική χρήση των δεδομένων σας και με
                                την επιλογή "Νέο Παιχνίδι" διαγράφονται αυτόματα.
210                         </div>
211                     </div>
212                 </section>
213             </main>
214             </div>
215
```

```
        <footer style="padding-top: 80px" class="text-center">
            <hr />
            <div class="container">
                <div class="row">
                    <div class="col-12">

                        <img class="rounded mr-5" alt="built on blockly" src="images/logo_built_on.
                        png" style='width: 200px;' data-holder-rendered="true">

                        <img class="rounded mt-3" alt="reset" src="images/title.png"
                        data-holder-rendered="true">

                    </div>
                </div>
            </div>
            <hr />
        </footer>

        <footer class="text-center">
            <div class="container">
                <div class="row">
                    <div class="col-12">
                        <p>Copyright ©2021 mmousiou. All rights reserved.</p>
                    </div>
                </div>
            </div>
        </footer>

        <!-- jQuery (necessary for Bootstrap's JavaScript plugins) -->
        <script src="maze/js/jquery-3.3.1.min.js"></script>
        <!-- Include all compiled plugins (below), or include individual files as needed -->
        <script src="maze/js/popper.min.js"></script>
        <script src="maze/js/bootstrap-4.3.1.js"></script>
</body>

</html>
```

Β9. Κώδικας σελίδας results-el.html

```
<!DOCTYPE html>
<html lang="el">

<head>
    <meta charset="utf-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />

    <!-- Bootstrap -->
    <link href="css/bootstrap-4.3.1.css" rel="stylesheet" />
    <link rel="stylesheet" href="maze/results.css" />

    <title>Αποτελέσματα</title>
</head>

<body>
    <!-- navbar -->
    <nav class="navbar navbar-expand-lg navbar-light bg-light">
        <a class="navbar-brand" href="about-el.html">
            <img src="images/logo_amazed.png" width="120">
        </a>
        <button class="navbar-toggler" type="button" data-toggle="collapse"
        data-target="#navbarSupportedContent1" aria-controls="navbarSupportedContent1"
        aria-expanded="false" aria-label="Toggle navigation"> <span class="navbar-toggler-icon"></
        span> </button>
        <div class="collapse navbar-collapse" id="navbarSupportedContent1">
            <span class="navbar-text mr-2 text-right">Συμπληρώστε το ονοματεπώνυμο και τη διεύθυνση
            email σας.</span>
```

```
            <form class="form-inline d-flex input-group w-auto">
                <input type="text" class="form-control mr-sm-2" id="name" name="name"
                placeholder="Ονοματεπώνυμο">
                <input type="email" class="form-control mr-sm-2" id="myemail" name="email"
                placeholder="Email" aria-describedby="emailHelp">
                <span id="emailHelp" class="form-text text-muted" style="display: none;
                ">Πληκτρολογήστε μια έγκυρη διεύθυνση e-mail.</span>
                <button id="save-btn" class="btn btn-primary my-2 my-sm-0" type="submit">Αποθήκευση</
                button>
            </form>
        </div>
    </nav>


    <!-- main app -->
    <div class="container" id="app">
        <hr />
        <div class="row justify-content-center">
            <div class="col-10">
                <h1>Συγχαρητήρια! Ολοκληρώσατε το παιχνίδι!</h1>
            </div>
        </div>
        <hr />
        <div class="row">
            <div class="col-md-8 col-sm-12">
                <div class="media">
                    <img class="rounded-circle border-left border-right border-secondary mr-4"
                    alt="100x100" style="width: 80px; height: 80px; " src="maze/maze2.png" />
                    <div class="media-body">
                        <h3 class="mt-3">
                            Συνολικό σκορ: <span id="total-score"></span>
                        </h3>
                    </div>
                </div>
            </div>
            <div class="col-md-4 col-sm-12">
                <div class="row justify-content-center">
                    <div>
                        <p>Διαγραφή δεδομένων και νέο παιχνίδι</p>
                        <div class="row justify-content-center">
                            <button id="clear" class="btn btn-primary">Νέο Παιχνίδι</button>
                        </div>
                    </div>
                </div>
            </div>


            <hr />

        </div>
        <hr />



        <hr />

        <div class="container text-lg-center">

            <h4>Αποτελέσματα</h4>
            <hr />

            <br />

            <table id='myTable'>
                <caption id="caption--text">
                    caption
                </caption>
                <thead>
                    <tr>
                        <th> Επίπεδο </th>
                        <th> Αποτέλεσμα </th>
                        <th> Βαθμολογία </th>
```

```
                <th> Χρόνος </th>
                <th> Πλήθος play </th>
                <th> Κώδικας JavaScript </th>
            </tr>
        </thead>
        <tbody>
            <tr>
                <td><b>1<b/></td>
                <td><span class="value" id="result-1">0</span></td>
                <td><span class="value" id="score-1">0</span></td>
                <td><span class="value" id="time-1">0</span></td>
                <td><span class="value" id="play-1">0</span></td>
                <td ><div class='code' id="code-1">  </div></td>
            </tr>
            <tr>
                <td ><b>2<b/></td>
                <td><span class="value" id="result-2">0</span></td>
                <td><span class="value" id="score-2">0</span></td>
                <td><span class="value" id="time-2">0</span></td>
                <td><span class="value" id="play-2">0</span></td>
                <td ><div class='code' id="code-2">  </div></td>
            </tr>
            <tr>
                <td ><b>3<b/></td>
                <td><span class="value" id="result-3">0</span></td>
                <td><span class="value" id="score-3">0</span></td>
                <td><span class="value" id="time-3">0</span></td>
                <td><span class="value" id="play-3">0</span></td>
                <td ><div class='code' id="code-3">  </div></td>
            </tr>
            <tr>
                <td ><b>4<b/></td>
                <td ><span class="value" id="result-4">0</span></td>
                <td><span class="value" id="score-4">0</span></td>
                <td><span class="value" id="time-4">0</span></td>
                <td><span class="value" id="play-4">0</span></td>
                <td ><div class='code' id="code-4">  </div></td>
            </tr>
            <tr>
                <td ><b>5<b/></td>
                <td><span class="value" id="result-5">0</span></td>
                <td><span class="value" id="score-5">0</span></td>
                <td><span class="value" id="time-5">0</span></td>
                <td><span class="value" id="play-5">0</span></td>
                <td ><div class='code' id="code-5">  </div></td>
            </tr>
            <tr>
                <td ><b>6<b/></td>
                <td><span class="value" id="result-6">0</span></td>
                <td><span class="value" id="score-6">0</span></td>
                <td><span class="value" id="time-6">0</span></td>
                <td><span class="value" id="play-6">0</span></td>
                <td ><div class='code' id="code-6">  </div></td>
            </tr>
            <tr>
                <td ><b>7<b/></td>
                <td><span class="value" id="result-7">0</span></td>
                <td><span class="value" id="score-7">0</span></td>
                <td><span class="value" id="time-7">0</span></td>
                <td><span class="value" id="play-7">0</span></td>
                <td ><div class='code' id="code-7">  </div></td>
            </tr>
            <tr>
                <td ><b>8<b/></td>
                <td><span class="value" id="result-8">0</span></td>
                <td><span class="value" id="score-8">0</span></td>
                <td><span class="value" id="time-8">0</span></td>
                <td><span class="value" id="play-8">0</span></td>
                <td ><div class='code' id="code-8">  </div></td>
            </tr>
```

```
                    <tr>
                        <td ><b>9<b/></td>
                        <td><span class="value" id="result-9">0</span></td>
                        <td><span class="value" id="score-9">0</span></td>
                        <td><span class="value" id="time-9">0</span></td>
                        <td><span class="value" id="play-9">0</span></td>
                        <td ><div class='code' id="code-9">  </div></td>
                    </tr>
                    <tr>
                        <td ><b>10<b/></td>
                        <td><span class="value" id="result-10">0</span></td>
                        <td><span class="value" id="score-10">0</span></td>
                        <td><span class="value" id="time-10">0</span></td>
                        <td><span class="value" id="play-10">0</span></td>
                        <td ><div class='code' id="code-10">  </div></td>
                    </tr>
                </tbody>
            </table>
            <br />

                    <button id="pdfExport" value="Export" onclick="Export()" class="btn
                    btn-primary mr-2">
                        Λήψη pdf
                    </button>

                <button id="xlsExport" value="Export" onclick="exportTableToExcel('myTable',
                'user-data')" class="btn btn-primary">
                    Λήψη excel
                </button>

        </div>
    </div>

      <hr />


      <footer class="text-center">
        <div class="container">
          <div class="row">
            <div class="col-12">
              <p>Copyright ©2021 mmousiou. All rights reserved.</p>
            </div>
          </div>
        </div>
      </footer>

    </div>
    <script src="maze/js/result.js"></script>
    <!-- jQuery (necessary for Bootstrap's JavaScript plugins) -->
    <script src="maze/js/jquery-3.3.1.min.js"></script>
    <!-- Include all compiled plugins (below), or include individual files as needed -->
    <script src="maze/js/popper.min.js"></script>
    <script src="maze/js/bootstrap-4.3.1.js"></script>

    <script type="text/javascript" src="https://cdnjs.cloudflare.com/ajax/libs/pdfmake/0.1.22/pdfmake.
    min.js"></script>
    <script type="text/javascript" src="https://cdnjs.cloudflare.com/ajax/libs/html2canvas/0.4.1/
    html2canvas.min.js"></script>

    <script type="text/javascript">
        function Export() {
            html2canvas(document.getElementById("myTable"), {
                onrendered: function(canvas) {
                    var data = canvas.toDataURL();
                    var docDefinition = {
                        content: [{
                            image: data,
                            width: 350,
                        }, ],
                    };
                    pdfMake.createPdf(docDefinition).download("data.pdf");
                },
            });
        }
    </script>
</body>
</html>
```

Β10. Αποθήκευση του πίνακα αποτελεσμάτων σε .xls και pdf μορφή

Δυο συναρτήσεις χρησιμοποιήθηκαν για την αποθήκευση του πίνακα αποτελεσμάτων

- η Export() της οποίας ο κώδικας είναι μέσα στο αρχείο results.html στις γραμμές 215-230 και ενεργοποιείται με on-click event στο πλήκτρο με αναγνωριστικό pdfExport (γραμμή 181) και
- η συνάρτηση exportTableToExcel() ο κώδικας της οποίας βρίσκεται στο result.js και ενεργοποιείται με on-click event στο πλήκτρο με αναγνωριστικό xlsExport (γραμμή 185 αρχείου results.html).

Ακολουθεί ο κώδικας της exportTableToExcel().

```
155  function exportTableToExcel(tableID, filename = '') {
156      var downloadLink;
157      var dataType = 'application/vnd.ms-excel';
158      var tableSelect = document.getElementById(tableID);
159      var tableHTML = tableSelect.outerHTML.replace(/ /g, '%20');
160
161      // Specify file name
162      filename = filename ? filename + '.xls' : 'excel_data.xls';
163
164      // Create download link element
165      downloadLink = document.createElement("a");
166
167      document.body.appendChild(downloadLink);
168
169      if (navigator.msSaveOrOpenBlob) {
170          var blob = new Blob(['\ufeff', tableHTML], {
171              type: dataType
172          });
173          navigator.msSaveOrOpenBlob(blob, filename);
174      } else {
175          // Create a link to the file
176          downloadLink.href = 'data:' + dataType + ', ' + tableHTML;
177
178          // Setting the file name
179          downloadLink.download = filename;
180
181          //triggering the function
182          downloadLink.click();
183      }
184  }
```

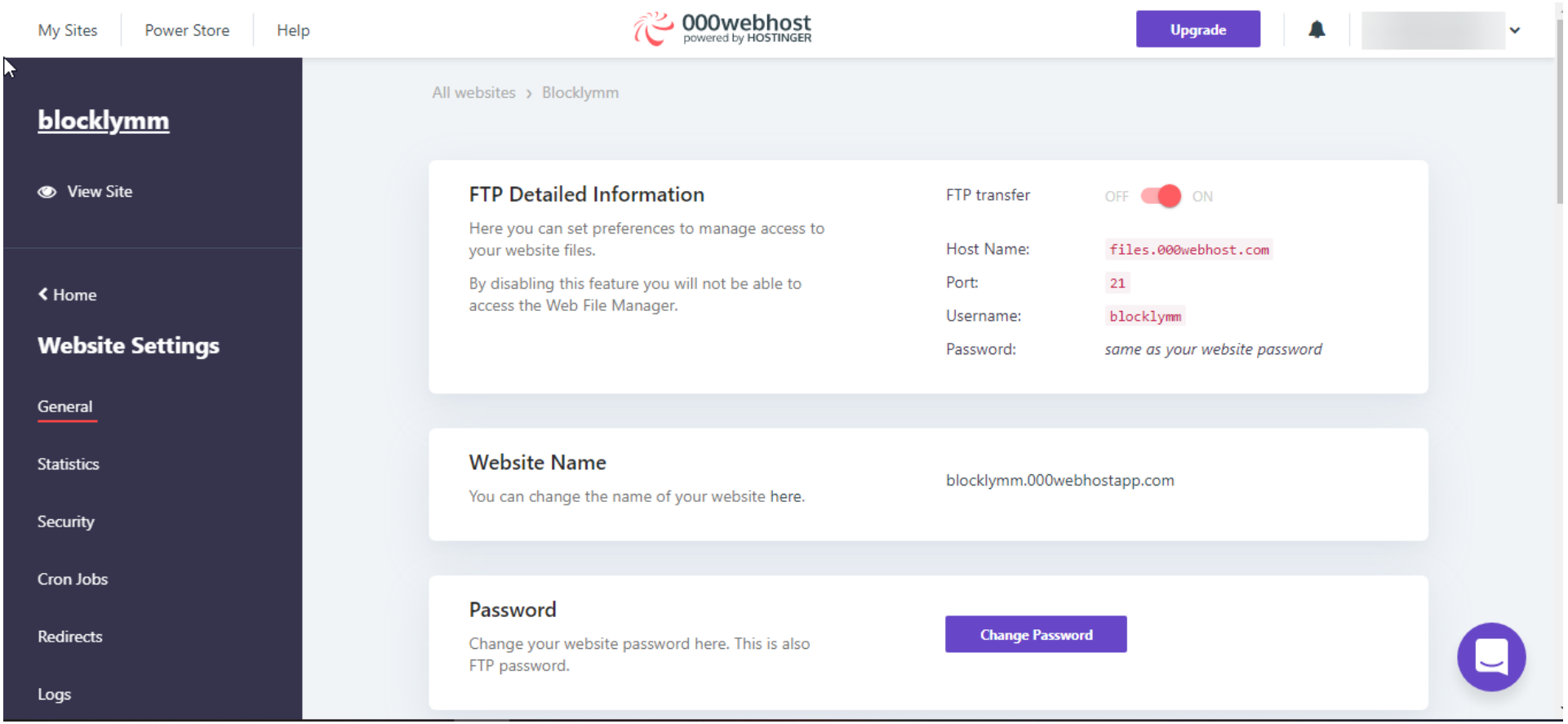
Παράρτημα Γ: Δημοσίευση παιχνιδιού στο διαδίκτυο

Ένα από τα ζητούμενα της εργασίας είναι η δημοσίευση του παιχνιδιού στο διαδίκτυο μέσω ενός δημόσιου διακομιστή (web server). Για να μείνουμε στην φιλοσοφία των δωρεάν προγραμμάτων και του ανοιχτού κώδικα επιλέχθηκε ο πάροχος φιλοξενίας Hostinger και η δωρεάν φιλοξενία που παρέχει σε ιστοσελίδες μέσω του 000webhost.com.
Η δημιουργία λογαριασμού είναι πολύ εύκολη με τη χρήση email, password ή με σύνδεση μέσω Facebook ή Google. Το επόμενο βήμα είναι η ονομασία της ιστοσελίδας και η ανάθεση κωδικού που θα χρειαστεί και για το ανέβασμα των αρχείων στον διακομιστή είτε από το file manager που παρέχεται στον χρήστη είτε με απομακρυσμένη σύνδεση του διακομιστή.
Με το ελεύθερο λογισμικό FileZilla συνδεθήκαμε στον απομακρυσμένο διακομιστή και ανεβάσαμε όλα τα αρχεία του φακέλου appengine που αφορούν τον λαβύρινθο.

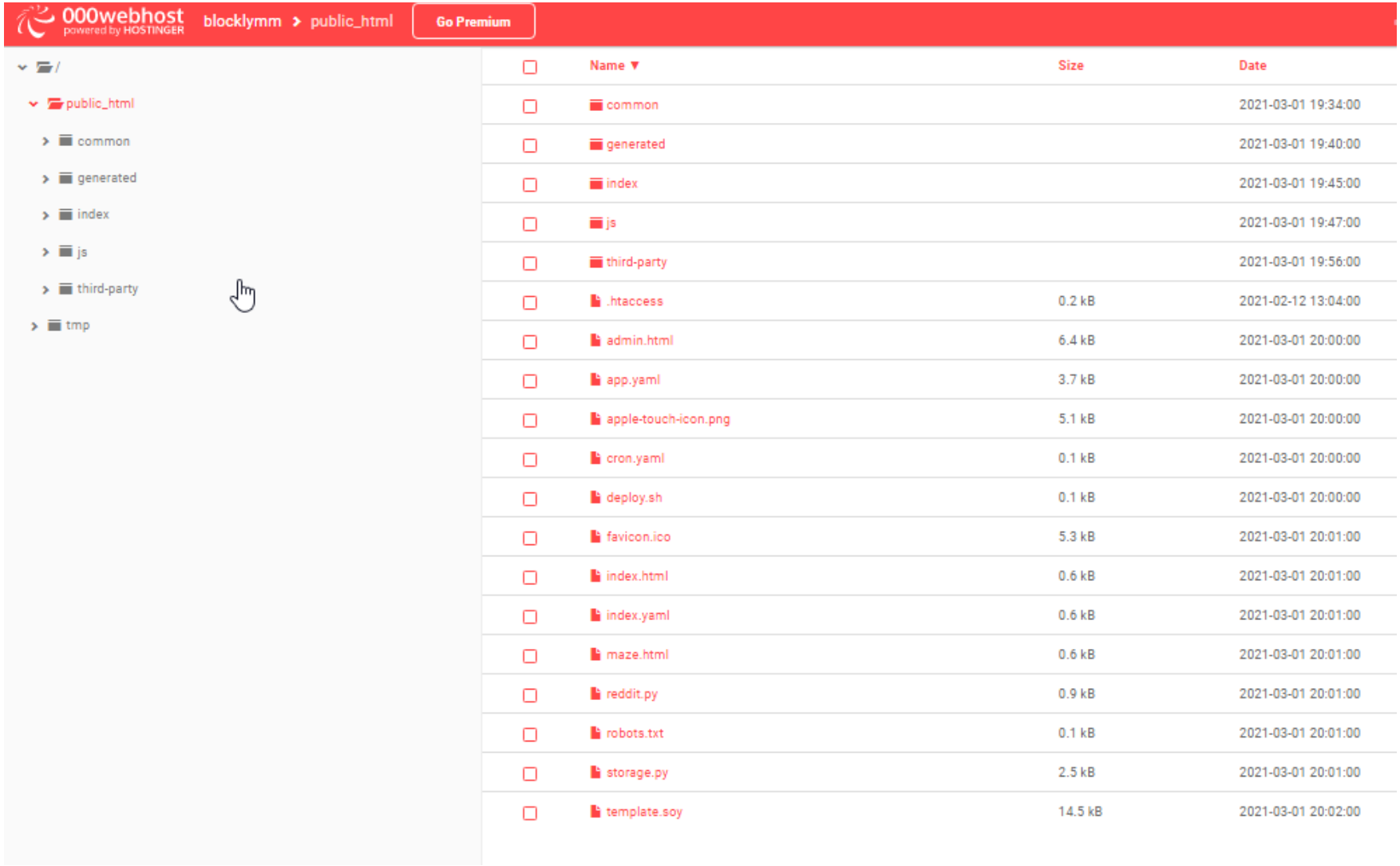
Οι σύνδεσμοι για τις δυο εκδόσεις του aMazeD είναι :
έκδοση 1: https://mazegamemm.000webhostapp.com/maze.html
έκδοση 2: https://mazenocomments.000webhostapp.com/maze.html

Ο κώδικας του παιχνιδιού είναι διαθέσιμος στο GitHub στους συνδέσμους:
έκδοση 1: https://github.com/mmousiou/aMazeD
έκδοση 2: https://github.com/mmousiou/aMazeD2

Εικόνα 50: Επιλογές 000webhost - στοιχεία για σύνδεση με πρωτόκολλο fttp



Εικόνα 51: Αρχεία ιστοσελίδας στον διακομιστή



Υπεύθυνη Δήλωση Συγγραφέα:

Δηλώνω ρητά ότι, σύμφωνα με το άρθρο 8 του Ν. 1599/1986 και τα άρθρα 2,4,6 παρ. 3 του Ν. 1256/1982, η παρούσα εργασία αποτελεί αποκλειστικά προϊόν προσωπικής εργασίας και δεν προσβάλλει κάθε μορφής πνευματικά δικαιώματα τρίτων και δεν είναι προϊόν μερικής ή ολικής αντιγραφής, οι πηγές δε που χρησιμοποιήθηκαν περιορίζονται στις βιβλιογραφικές αναφορές και μόνον.