ΠΑΝΕΠΙΣΤΗΜΙΟ ΙΩΑΝΝΙΝΩΝ

ΣΧΟΛΗ ΕΠΙΣΤΗΜΩΝ ΤΗΣ ΑΓΩΓΗΣ

ΠΑΙΔΑΓΩΓΙΚΟ ΤΜΗΜΑ ΔΗΜΟΤΙΚΗΣ ΕΚΠΑΙΔΕΥΣΗΣ

ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ

«ΔΙΔΑΚΤΙΚΗ ΚΑΙ ΤΕΧΝΟΛΟΓΙΕΣ ΜΑΘΗΣΗΣ ΤΩΝ ΦΥΣΙΚΩΝ ΕΠΙΣΤΗΜΩΝ»

ΤΙΤΛΟΣ ΔΙΠΛΩΜΑΤΙΚΗΣ ΕΡΓΑΣΙΑΣ

Παράλληλες διεργασίες στην υπολογιστική σκέψη μαθητών

Μπέκος Δημήτριος ΑΜ:18

Επιβλέπων Καθηγητής: Μικρόπουλος Αναστάσιος

ΙΩΑΝΝΙΝΑ 2022

Στην μνήμη του πολυαγαπημένου μου πατέρα

Απόστολου Μπέκου

ΕΥΧΑΡΙΣΤΙΕΣ

Για την πραγματοποίηση την παρούσας διπλωματικής εργασίας ευχαριστώ τον καθηγητή του Παιδαγωγικού Τμήματος Δημοτικής Εκπαίδευσης του Πανεπιστήμιου Ιωαννίνων και επιβλέποντα της μελέτης μου κ. Μικρόπουλο Αναστάσιο, για την συνεργασία, την καθοδήγηση και την πολύτιμη βοήθεια που μου παρείχε κατά τη διάρκεια της εκπόνησης της.

Ευχαριστώ επίσης, τον σχολικό σύμβουλο στον κλάδο της Πληροφορικής κ. Λαδιά Αναστάσιο για την δικιά του πολύτιμη βοήθεια και καθοδήγηση σε όλα τα στάδια της εργασίας, τις κατευθυντήριες συμβουλές, τον χρόνο που αφιέρωσε καθώς και για την παροχή του δείγματος της παρούσας μελέτης.

Ευχαριστίες απευθύνω και στα μέλη τριμελούς εξεταστικής επιτροπής της μεταπτυχιακής διπλωματικής εργασίας, αναπληρωτές καθηγητές του Παιδαγωγικού Τμήματος Δημοτικής Εκπαίδευσης του Πανεπιστήμιου Ιωαννίνων κυρίους Δημήτριο Μαυρίδη και Κωνσταντίνο Τάτση για τις εποικοδομητικές τους επισημάνσεις.

Τέλος, ευχαριστώ τους φίλους, τις φίλες και την μητέρα μου Κωνσταντίνα για την ηθική τους υποστήριξη.

ΠΕΡΙΕΧΟΜΕΝΑ

ΕΥΧΑΡΙΣΤΙΕΣ 4

ΠΕΡΙΕΧΟΜΕΝΑ 5

ΕΥΡΕΤΗΡΙΟ ΣΧΗΜΑΤΩΝ 7

ΕΥΡΕΤΗΡΙΟ ΠΙΝΑΚΩΝ 9

ΠΕΡΙΛΗΨΗ 10

ABSTRACT 12

ΕΙΣΑΓΩΓΗ 13

1.ΘΕΩΡΗΤΙΚΟ ΠΛΑΙΣΙΟ 15

1.1. Υπολογιστική σκέψη 15

1.2. Προγραμματιστικά υποδείγματα 16

1.3. Παράλληλος και ψευδοπαράλληλος προγραμματισμός 20

1.3.1. Παράλληλος προγραμματισμός 20

1.3.2 Προβλήματα παράλληλου προγραμματισμού και τρόποι επίλυσης 22

1.3.3. Ψευδοπαράλληλος προγραμματισμός 23

1.4. Διδακτική και διδασκαλία του προγραμματισμού 24

1.5. Διδασκαλία του προγραμματισμού στο δημοτικό με Scratch 27

1.5.1. Διδασκαλία του προγραμματισμού στο δημοτικό σχολείο 27

1.5.2. Scratch 28

1.6. Βιβλιογραφική ανασκόπηση παράλληλου προγραμματισμού στο δημοτικό και τις πρώτες τάξεις γυμνασίου 31

1.6.1. Μεθοδολογία 31

1.6.2. Βιβλιογραφική ανασκόπηση 32

1.7. Κατηγοριοποίηση τοπολογιών παράλληλου προγραμματισμού στο Scratch με χρήση της ταξινομίας SOLO 37

1.7.1. Τοπολογίες παράλληλου προγραμματισμού στο Scratch 37

1.7.2. Ταξινομία SOLO 38

1.7.3. Τοπολογίες παράλληλου προγραμματισμού και επίπεδα της SOLO 39

2.ΕΡΕΥΝΗΤΙΚΟ ΜΕΡΟΣ 47

2.1 Μεθοδολογία 47

2.1.1. Σκοπός έρευνας 47

2.1.2. Ανάλυση δεδομένων 47

2.1.3. Δείγμα 48

2.1.3.1 ΚωδικΌραμα 48

2.1.4. Προσδιορισμός μονάδας Ανάλυσης 50

2.1.5. Κατηγοριοποίηση δεδομένων 52

2.2 Αποτελέσματα 53

2.3 Συζήτηση και συμπεράσματα 62

2.3.1. Περιορισμοί της έρευνας 71

2.3.2 Προτάσεις για μελλοντική έρευνα 71

Αναφορές 72

ΠΑΡΑΡΤΗΜΑ 77

ΕΥΡΕΤΗΡΙΟ ΣΧΗΜΑΤΩΝ

Σχήμα 1.3. 1 Μοντέλο Διασύνδεσης Μεταβίβασης Κοινοχρήστων Μηνυμάτων.....22
Σχήμα 1.3. 2 Μοντέλο συστημάτων πολυεπεξεργαστών κοινόχρηστης μνήμης 22
Σχήμα 1.5. 1 Η οθόνη του Scratch 30
Σχήμα 1.7. 1 Σειριακός κώδικας χωρίς παράλληλες διεργασίες.....40
Σχήμα 1.7. 2 Μη αιτιοκρατική συμπεριφορά παράλληλων διεργασιών.....40
Σχήμα 1.7. 3 Ανεξάρτητες παράλληλες διεργασίες με ταυτόχρονη εκκίνηση από την ίδια αιτία 41
Σχήμα 1.7. 4 Εξωτερικά συμβάντα ως αιτία πυροδότησης παράλληλων διεργασιών στο Scratch.....41
Σχήμα 1.7. 5 Εσωτερικά συμβάντα ως αιτία πυροδότησης παράλληλων διεργασιών στο Scratch.....42
Σχήμα 1.7. 6 Δημιουργία κλώνων ως αιτία πυροδότησης παράλληλων διεργασιών στο Scratch.....42
Σχήμα 1.7. 7 Ανεξάρτητες παράλληλες διεργασίες με μη ταυτόχρονη πυροδότηση από διαφορετική αιτία.....43
Σχήμα 1.7. 8 Διαδοχική ενεργοποίηση διεργασιών με μη ταυτόχρονη πυροδότηση από διαφορετική αιτία.....43
Σχήμα 1.7. 9 (α) Παράλληλες διεργασίες σε μόνιμη κατάσταση εκτέλεσης - (β) Παράλληλες διεργασίες σε κατάσταση επαγρύπνησης.....43
Σχήμα 1.7. 10 Πατρική διεργασία συνεχίζει να εκτελείται παράλληλα με τις διεργασίες απογόνους μετά την πυροδότηση τους 44
Σχήμα 1.7. 11 Σύγχρονη επικοινωνία διεργασιών 44
Σχήμα 1.7. 12 Ασύγχρονη επικοινωνία διεργασιών.....45
Σχήμα 1.7. 13 Έλεγχος πρόσβασης σε κρίσιμη περιοχή κώδικα από πολλαπλές διεργασίες με τη χρήση σηματωρών 46
Σχήμα 2.1. 1 Ένα παράδειγμα ολοκληρωμένου ΚωδικΟράματος 50
Σχήμα 2.1. 2 Ένα παράδειγμα κελιού ΚωδικΟράματος που αποτελεί μία μονάδα ανάλυσης 51
Σχήμα 2.1. 3 Μια ολόκληρη γραμμή ΚωδικΟράματος που δείχνει ποια παράλληλα σενάρια εκτελούνται.....51

Σχήμα 2.2. 1 Συχνότητα κωδικΟραμάτων για κάθε κατηγορία της SOLO 57
Σχήμα 2.2. 2 Ποσοστό κωδικΟραμάτων για κάθε κατηγορία της SOLO.....58
Σχήμα 2.2. 3 Μέσος όρος για το μέγεθος των κωδικΟραμάτων κάθε επιπέδου της ταξινομίας SOLO 60
Σχήμα 2.2. 4 Μέσος όρος για το πλήθος των καταστάσεων των κωδικΟραμάτων κάθε επιπέδου της ταξινομίας SOLO.....61
Σχήμα 2.2. 5 Μέσος όρος για το πλήθος των διαδικασιών των κωδικΟραμάτων κάθε επιπέδου της ταξινομίας SOLO.....61
Σχήμα 2.2. 6 Μέσος όρος για το πλήθος κελιών με παράλληλα σενάρια του ίδιου αντικειμένου των κωδικΟραμάτων κάθε επιπέδου της ταξινομίας SOLO 62
Σχήμα 2.3. 1 Στιγμιότυπο του ΚωδικΟράματος 24 που η ίδια αιτία πυροδοτεί την ταυτόχρονη εκκίνηση παράλληλων σεναρίων.....63
Σχήμα 2.3. 2 Στιγμιότυπο του ΚωδικΟράματος 2 που δείχνει ένα τμήμα κώδικα σε μόνιμη κατάσταση επανάληψης 63
Σχήμα 2.3. 3 Στιγμιότυπο του ΚωδικΟράματος 13 που διαφορετική αιτία πυροδοτεί την εκκίνηση παράλληλων σεναρίων 65
Σχήμα 2.3. 4 Στιγμιότυπο του ΚωδικΟράματος 6 κατά το οποίο το πατρικό σενάριο συνεχίζει να εκτελείται παράλληλα με το σενάριο απόγονο 66
Σχήμα 2.3. 5 Στιγμιότυπο του ΚωδικΟράματος 18 που γίνεται χρήση της εντολής “Όταν” και όχι της “Αν” 66
Σχήμα 2.3. 6 Στιγμιότυπο του ΚωδικΟράματος 3 που γίνεται χρήση της εντολής “Περίμενε x δευτερόλεπτα”.....67
Σχήμα 2.3. 7 Στιγμιότυπο του ΚωδικΟράματος 4 που επιτυγχάνεται ασύγχρονη επικοινωνία με τη χρήση μεταβλητής.....68
Σχήμα 2.3. 8 Στιγμιότυπο του ΚωδικΟράματος 17 που επιτυγχάνεται σύγχρονη επικοινωνία με χρήση της εντολής “Μετέδωσε και Περίμενε”.....68

ΕΥΡΕΤΗΡΙΟ ΠΙΝΑΚΩΝ

Πίνακας 2. 1 Τοπολογίες παράλληλου προγραμματισμού στα ΚωδικΌραματα 1-25.....55

Πίνακας 2. 2 Συχνότητα και ποσοστό κωδικΟραμάτων για κάθε κατηγορία της SOLO.....57

Πίνακας 2. 3 Δεδομένα για το μέγεθος, το πλήθος καταστάσεων, το πλήθος διαδικασιών και το πλήθος κελιών με παράλληλα σενάρια του ίδιου αντικειμένου για τα κωδικΌραματα 1 έως 25 59

Πίνακας 2. 4 Μέσος όρος για το μέγεθος, το πλήθος καταστάσεων, το πλήθος διαδικασιών και το πλήθος κελιών με παράλληλα σενάρια του ίδιου αντικειμένου, των κωδικΟραμάτων κάθε επιπέδου της ταξινομίας SOLO.....60

ΠΕΡΙΛΗΨΗ

Η παρούσα μεταπτυχιακή διπλωματική εργασία αφορά τον παράλληλο προγραμματισμό, δηλαδή την εκτέλεση δύο ή περισσότερων εντολών προγραμματιστικού κώδικα ταυτόχρονα. Πιο συγκεκριμένα, η μελέτη έχει ως στόχο την αξιολόγηση της χρήσης παράλληλου προγραμματισμού σε προγράμματα ανεπτυγμένα από μαθητές που βρίσκονται στις τελευταίες τάξεις της πρωτοβάθμιας εκπαίδευσης.

Για την διεκπεραίωση του στόχου της έρευνας, 25 κώδικες μαθητών ηλικίας δέκα έως δεκατριών ετών, αξιολογήθηκαν με κριτήριο την παραλληλία. Οι κώδικες αυτοί είναι γραμμένοι στο περιβάλλον οπτικού προγραμματισμού Scratch, και αναπτύχθηκαν από τους μαθητές για να ελέγξουν ρομποτικές συσκευές. Το Scratch υποστηρίζει το προγραμματιστικό υπόδειγμα του παράλληλου προγραμματισμού καθώς σε αυτό το περιβάλλον μπορούν να εκτελούνται ταυτόχρονα πάνω από ένα τμήματα κώδικα (σενάρια), με αποτέλεσμα συχνά να εντοπίζονται στοιχεία παραλληλίας σε κώδικες μαθητών, ακόμη κι αν αυτοί δεν έχουν διδαχθεί κάτι ανάλογο στο πλαίσιο του σχολικού αναλυτικού προγράμματος τους. Η αξιολόγηση πραγματοποιήθηκε με την ένταξη κάθε κώδικα στα πέντε επίπεδα της ταξινομίας SOLO (Structure of Observed Learning Outcomes), η οποία αξιολογεί το γνωστικό επίπεδο του μαθητή ταξινομώντας τη γνώση σε επίπεδα πολυπλοκότητας παρατηρώντας τα μαθησιακά αποτελέσματα του. Στην παρούσα εργασία η ένταξη των κωδίκων στα πέντε επίπεδα της ταξινομίας γίνεται με βάση τις τοπολογίες παράλληλου προγραμματισμού που παρατηρούνται σε καθέναν από αυτούς.

Τα αποτελέσματα της έρευνας έδειξαν πως η πλειοψηφία των μαθητών χρησιμοποιεί τοπολογίες παράλληλου προγραμματισμού μεσαίας πολυπλοκότητας και αναπτύσσει προγραμματιστικούς κώδικες που εντάσσονται κυρίως στο τρίτο από τα πέντε επίπεδα της ταξινομίας SOLO. Πιο συγκεκριμένα, κατά κύριο λόγο οι μαθητές προγραμματίζουν σενάρια που τρέχουν παράλληλα, πυροδοτούνται από διαφορετικές αιτίες και λειτουργούν ανεξάρτητα καθώς δεν επικοινωνούν μεταξύ τους.

Από τα αποτελέσματα αυτά αναδεικνύονται οι παράλληλες διεργασίες στην υπολογιστική σκέψη των μαθητών. Οι νεαροί προγραμματιστές, που δεν έχουν διεισδύσει στο σειριακό προγραμματισμό, σκέφτονται παράλληλα καθώς και στον πραγματικό κόσμο οι διάφορες οντότητες λειτουργούν ταυτόχρονα. Επίσης, από τα αποτελέσματα της εργασίας προκύπτει

και η δυναμική για την ένταξη του παράλληλου προγραμματισμού στο αναλυτικό πρόγραμμα στις τελευταίες τάξεις του δημοτικού σχολείου.

Λέξεις κλειδιά: Παράλληλος Προγραμματισμός, Παράλληλες Διεργασίες, Υπολογιστική σκέψη, Δημοτικό, Ταξινομία SOLO, Scratch

ABSTRACT

This postgraduate study is about parallel programming, namely the execution of two or more programming commands simultaneously. More specifically, the research aims to evaluate the use of parallel programming that students make in the last grades of primary education in their programming codes.

To accomplish the goal of the research 25 codes of students aged ten to thirteen, were evaluated with the criterion of parallelism. These codes are written in the Scratch visual programming environment and were developed by students to control robotic devices. Scratch supports the programming model of parallel programming as in this environment, more than one blocks of code (scripts) can be executed at the same time so that parallel elements are often found in student codes, even if they have not been taught something similar in their curricula. The assessment was performed by integrating each code into the five levels of the SOLO (Structure of Observed Learning Outcomes) taxonomy, which evaluates the student's cognitive level of complexity by classifying knowledge into levels of complexity by observing learning outcome. In the present work, the integration of the codes in the five levels of the taxonomy is done based on the parallel programming topologies that are observed in each of them.

The results of the research showed that most students use parallel programming topologies of medium complexity and develop programming codes that are mainly categorized to the third of the five levels of SOLO taxonomy. More specifically, the students mainly program scripts that run in parallel, are triggered by different causes and operate independently as they do not communicate with each other.

These results highlight the parallel processes in students computational thinking. Young developers, who have not delved into serial programming, think in parallel as in the real world the various entities operate "in parallel". Also, the results of work show the potential for the integration of parallel programming in the curricula at the last grades of primary education.

Key words: Parallel Programming, Concurrent Processes, Computational Thinking, Primary Education, SOLO taxonomy, Scratch

ΕΙΣΑΓΩΓΗ

Τα τελευταία χρόνια οι ηλεκτρονικοί υπολογιστές αυξάνουν συνεχώς τις δυνατότητές τους για παράλληλη επεξεργασία και υπάρχει ανάγκη για προγραμματιστές που θα σχεδιάζουν παράλληλο κώδικα. Ένα παράλληλο πρόγραμμα αποτελείται από δύο ή περισσότερα τμήματα κώδικα που εκτελούνται ταυτόχρονα, αντίθετα με το σειριακό προγραμματιστικό μοντέλο κατά το οποίο, σε οποιαδήποτε χρονική στιγμή εκτελείται μόνο μια εντολή (Δημακόπουλος, 2015). Η διδακτική του παράλληλου προγραμματισμού από μικρή ηλικία αποτελεί ένα πρώτο βήμα προς αυτή την κατεύθυνση για τους προγραμματιστές του μέλλοντος (Gregg, Tychonievich, Cohoon, & Hazelwood 2012). Η ανάγκη για ένταξη της διδακτικής του παράλληλου προγραμματισμού στο γενικότερο κομμάτι της διδακτικής του προγραμματισμού στην πρωτοβάθμια εκπαίδευση, φέρνει στην επιφάνεια την ανάγκη για αξιολόγηση των δεξιοτήτων παράλληλης επεξεργασίας από μαθητές δημοτικού. Η παρούσα διπλωματική εργασία προτείνει την αξιολόγηση κώδικα μαθητών δημοτικού, με τη χρήση της ταξινομία SOLO και του προγραμματιστικού περιβάλλοντος Scratch, στο πρότυπο που προτείνουν οι Λαδιάς & Καρβουνίδης το 2021.

Το πρώτο κεφάλαιο της εργασίας αφορά το θεωρητικό πλαίσιο της. Αρχικά, γίνεται αποσαφήνιση του όρου της υπολογιστικής σκέψης. Επίσης, παρουσιάζονται τα διάφορα προγραμματιστικά υποδείγματα και ειδικότερα αυτά του παράλληλου και ψευδοπαράλληλου προγραμματισμού. Στη συνέχεια, γίνεται περιγραφή των μεθόδων της διδακτικής προγραμματισμού αλλά και πιο συγκεκριμένα της διδακτικής του προγραμματισμού στη βαθμίδα του δημοτικού με τη χρήση της πλατφόρμας οπτικού προγραμματισμού Scratch. Επιπλέον, το κεφάλαιο αυτό περιλαμβάνει την βιβλιογραφική ανασκόπηση που πραγματοποιήθηκε και που αφορά τον παράλληλο προγραμματισμό και τη χρήση του από μαθητές δημοτικού και γυμνασίου. Στο τέλος του πρώτου κεφαλαίου περιγράφεται η κατηγοριοποίηση των τοπολογιών παράλληλου προγραμματισμού στο Scratch με χρήση της ταξινομίας SOLO.

Το δεύτερο κεφάλαιο της εργασίας αφορά το ερευνητικό κομμάτι της. Αρχικά, περιγράφεται η μεθοδολογία της εργασίας δηλαδή ο σκοπός της έρευνας, η ερευνητική υπόθεση με βάση την βιβλιογραφία, το δείγμα που χρησιμοποιήθηκε, ο προσδιορισμός της μονάδας ανάλυσης και η μέθοδος κατηγοριοποίησης των δεδομένων. Στη συνέχεια, ακολουθεί η αναλυτική

παρουσίαση των αποτελεσμάτων. Τέλος, παρουσιάζεται η ερμηνεία των αποτελεσμάτων, η εξαγωγή συμπερασμάτων, οι περιορισμοί της έρευνας και οι προτάσεις για μελλοντική έρευνα.

1.ΘΕΩΡΗΤΙΚΟ ΠΛΑΙΣΙΟ

1.1. Υπολογιστική σκέψη

Σύμφωνα με την Wing (2006), η υπολογιστική σκέψη αφορά την επίλυση προβλημάτων, τον σχεδιασμό συστημάτων και την κατανόηση της ανθρώπινης συμπεριφοράς, αντλώντας από τις βασικές έννοιες της επιστήμης των υπολογιστών. Ο Aho (2012) αναφέρει ότι η υπολογιστική σκέψη αφορά τις διαδικασίες σκέψεις που εμπλέκονται στη διαμόρφωση των προβλημάτων ώστε να μπορούν να αναπαρασταθούν μέσω υπολογιστικών βημάτων και αλγορίθμων. Σημαντικό μέρος αυτής της διαδικασίας αποτελεί η εύρεση κατάλληλων μοντέλων υπολογισμού μέσω των οποίων διαμορφώνεται το πρόβλημα και προκύπτουν οι λύσεις του. Ένας πιο πρόσφατος ορισμός προσεγγίζει την υπολογιστική σκέψη ως το εννοιολογικό θεμέλιο που απαιτείται για την επίλυση προβλημάτων αποτελεσματικά και αποδοτικά, δηλαδή αλγοριθμικά, με ή χωρίς τη βοήθεια υπολογιστών με λύσεις που είναι επαναχρησιμοποιήσιμες σε διαφορετικά περιβάλλοντα (Shute, Sun, & Asbell-Clarke, 2017).

Η υπολογιστική σκέψη περιγράφεται από μια πληθώρα χαρακτηριστικών. Αρχικά, περιλαμβάνει την αναδιαμόρφωση ενός φαινομενικά δύσκολου προβλήματος σε ένα άλλο πρόβλημα που μπορεί να λυθεί διαιρώντας το σε μικρότερα προβλήματα ή μετασχηματίζοντάς το (Wing, 2006). Επίσης, περιλαμβάνει την ικανότητα για κατασκευή ενός μεγάλου συστήματος, βασισμένη σε προϋπάρχουσες πληροφορίες. Η χρησιμοποίηση και ο μετασχηματισμός του συστήματος αυτού, δεν έχει ως προϋπόθεση την πλήρη κατανόηση κάθε λεπτομέρειάς του. Επιπρόσθετα, περιλαμβάνει την ικανότητα για αφαίρεση, δηλαδή την αναγνώριση και την άντληση των σημαντικών στοιχείων ενός προβλήματος ή συστήματος. Επίσης, αφορά τη λήψη σκόπιμων ενεργειών για την προσέγγιση λύσεων. Η υπολογιστική σκέψη αποτελεί βασική δεξιότητα που πρέπει ο άνθρωπος να κατέχει για να λειτουργεί στην σημερινή κοινωνία. Ουσιαστικά, είναι ο τρόπος σκέψης του ανθρώπου και όχι του υπολογιστή και αποτελεί αναπόσπαστο κομμάτι στην πραγματικότητα της ανθρωπότητας. Η στρατηγική για την νίκη σε ένα παιχνίδι και η τοποθέτηση με ένα αντιπαράδειγμα σε μια λεκτική διαφωνία, είναι χαρακτηριστικά παραδείγματα της υπολογιστικής σκέψης στην καθημερινότητα του ανθρώπου (Wing, 2006). Συμπληρωματικά, κατά την Wing (2008), η υπολογιστική σκέψη είναι μια μορφή αναλυτικής σκέψης που μοιράζεται με τη μαθηματική σκέψη τις γενικές μεθόδους για την επίλυση ενός

προβλήματος, με την μηχανική σκέψη τις μεθόδους σχεδίασης και αξιολόγησης ενός συστήματος που λειτουργεί στον πραγματικό κόσμο και με την επιστημονική σκέψη, την ευφυΐα και την ανθρώπινη συμπεριφορά.

Συχνά, η υπολογιστική σκέψη συνδέεται με την επιστήμη των υπολογιστών (Shute, Sun & Asbell-Clarke, 2017). Συγκεκριμένα, συνδέεται με τις τεχνικές επίλυσης προβλημάτων που εγείρονται στον χώρο της επιστήμης των υπολογιστών (Aho, 2012). Στο πλαίσιο αυτό, η υπολογιστική σκέψη αφορά την αξιολόγηση κώδικα ενός προγράμματος όχι μόνο από την αποτελεσματικότητα και την ορθότητά του, αλλά και την αισθητική, την απλότητα και την κομψότητά του. Επιπρόσθετα, η υπολογιστική σκέψη περιλαμβάνει την αποφυγή των χειρότερων καταστάσεων, την διόρθωση πιθανών σφαλμάτων και την διαχείριση ανταγωνισμού για πόρους (Wing, 2006). Οι δεξιότητες υπολογιστικής σκέψης δεν ταυτίζονται με τις προγραμματιστικές δεξιότητες, αλλά όταν κάποιο άτομο προγραμματίζει, έχει όφελος από την χρήση της (Shute, Sun & Asbell-Clarke, 2017).

Για την καλλιέργεια της υπολογιστικής σκέψης στους μαθητές κάτω των δεκαοκτώ ετών έχει προταθεί η ένταξη των εννοιών της στα προγράμματα σπουδών (Lee et al., 2011). Οι μαθητές είναι σε θέση να χρησιμοποιούν την αφαίρεση, την ανάλυση και τα άλλα συστατικά της υπολογιστικής σκέψης, όταν τους παρέχεται πρόσβαση σε πλούσια μαθησιακά περιβάλλοντα που περιλαμβάνουν εξειδικευμένους εκπαιδευτικούς, αναπτυξιακά ζητήματα και νέες τεχνολογίες, όπως μοντελοποίηση, ρομποτική, προσομοιώσεις και ανάπτυξη παιχνιδιών. Επίσης, η ανάπτυξη της υπολογιστικής σκέψης είναι σημαντικό να συμβαίνει σε ένα χρονικό συνεχές και όχι σε διάσπαρτες χρονικές στιγμές. Για την ένταξη της υπολογιστικής σκέψης στις νεαρές ηλικίες απαιτούνται προσπάθειες προς δύο κατευθύνσεις (Barr & Stephenson, 2011). Αρχικά, προς στην υπερπήδηση των σημαντικών εμποδίων υποδομής και στη συνέχεια, προς στην παροχή κατάλληλων εκπαιδευτικών πόρων στους εκπαιδευτικούς των ηλικιακών αυτών ομάδων.

1.2. Προγραμματιστικά υποδείγματα

Τις τελευταίες δεκαετίες έχουν αναπτυχθεί πολλές γλώσσες προγραμματισμού και ένας τρόπος για την κατανόηση των ομοιοτήτων και των διαφορών τους γίνεται με τον διαχωρισμό των γλωσσών αυτών σε προγραμματιστικά υποδείγματα. Κατά τον Samuel (2017), οι πιο βασικές κατηγορίες των προγραμματιστικών υποδειγμάτων είναι το

προστακτικό, το συναρτησιακό, το αντικειμενοστραφές, το λογικό, το παράλληλο και το υπόδειγμα προγραμματισμού βασισμένο σε γεγονότα. Αυτά τα υποδείγματα φαίνεται να κυριαρχούν στην ανάπτυξη λογισμικού και πολλές γλώσσες προγραμματισμού συνδυάζουν στοιχεία από παραπάνω από ένα υποδείγματα (Samuel, 2017). Ο Δημακόπουλος (2015) αναφέρει πως κρίσιμο ρόλο τα επόμενα χρόνια θα παίξει το παράλληλο υπόδειγμα προγραμματισμού.

Οι προστακτικές γλώσσες προγραμματισμού έχουν βασικά χαρακτηριστικά που τις διαφοροποιούν από μία άλλη μεγάλη κατηγορία γλωσσών προγραμματισμού, τις δηλωτικές γλώσσες (Σακελλαρίου κ.α., 2015). Οι διαφορές μεταξύ δηλωτικών και προστακτικών γλωσσών προγραμματισμού εντοπίζονται στον τρόπο ορισμού του προβλήματος, στο ρόλο των μεταβλητών, στο μοντέλο διαχείρισης της μνήμης και στο πεδίο δράσης των μεταβλητών. Όσον αφορά τον ορισμό του προβλήματος, οι προστακτικές γλώσσες προγραμματισμού περιγράφουν την επίλυση ενός προβλήματος με τη χρήση αυστηρών ακολουθιών, δηλαδή το «πώς» το πρόγραμμα επιλύεται, ενώ στις δηλωτικές γλώσσες τα προγράμματα που αναπτύσσονται, ορίζουν σε υψηλό επίπεδο «τί» θα επιλυθεί. Οι μεταβλητές στο προστακτικό υπόδειγμα αλλάζουν τιμές κατά τη διάρκεια εκτέλεσης του προγράμματος ενώ στο δηλωτικό σε κάθε μεταβλητή εκχωρείται μία τιμή η οποία δεν αλλάζει κατά τη διάρκεια της εκτέλεσης. Επίσης, στις προστακτικές γλώσσες το πρόγραμμα μεταφέρει δεδομένα από το ένα τμήμα του στο άλλο, οπότε ο προγραμματιστής πρέπει να είναι προσεκτικός για να μην τροποποιηθούν δεδομένα από κάποιο τμήμα ενώ αυτά χρησιμοποιούνται σε κάποιο άλλο. Σε αντίθεση, οι δηλωτικές γλώσσες αποφεύγουν τον κίνδυνο επειδή δεν χρησιμοποιούν αποκλειστικά τη μνήμη ως ενδιάμεσο χώρο αποθήκευσης των δεδομένων. Τέλος, το πεδίο δράσης των μεταβλητών στις προστατικές γλώσσες, είναι ευρύ και συνήθως είναι ολόκληρο το πρόγραμμα ή ένα υποπρόγραμμα, ενώ στις δηλωτικές, η τοπικότητα των μεταβλητών βρίσκεται στο πλαίσιο κάθε μεμονωμένης φράσης του προγράμματος.

Στο προστακτικό υπόδειγμα προγραμματισμού ανήκει ο **δομημένος προγραμματισμός**. (Avacheva & Prutzkow, 2020). Υπάρχουν διαφωνίες για τον ορισμό του δομημένου προγραμματισμού αλλά δύο χαρακτηριστικά είναι αυτά που τον περιγράφουν με συνέπεια (Hunt, 1979). Το πρώτο είναι ότι, ένα δομημένο πρόγραμμα είναι μια ιεραρχική διάταξη, ανεξάρτητων μεταξύ τους συστατικών και το δεύτερο πως η ροή εντός του προγράμματος

γίνεται αποκλειστικά με τρεις μορφές ελέγχου, την επιλογή, την επανάληψη και την ακολουθία. Οι βασικές δομές που υποστηρίζουν αυτές τις μορφές ελέγχου είναι η δομή IF THEN ELSE για την επιλογή, η δομή DO WHILE για την επανάληψη και η απλή τοποθέτηση των εντολών μία μετά την άλλη για την συνέχεια.

Ο **διαδικαστικός προγραμματισμός** (procedural programming) ανήκει και αυτός στον χώρο του προστακτικού υποδείγματος προγραμματισμού (Avacheva & Prutzkow, 2020). Έχει ως βασικό στοιχεία τη δόμηση του προγράμματος και την επαναλαμβανόμενη χρήση υποπρογραμμάτων, τα οποία είτε επιτελούν εργασίες γενικής φύσης είτε απευθύνονται σε ένα τμήμα του συνολικού προβλήματος (Μαστοροκώστας, 2015). Ο διαδικαστικός προγραμματισμός είναι δομημένος προγραμματισμός, επεκταμένος με χρήση ρουτινών (Avacheva & Prutzkow, 2020). Κάθε ρουτίνα είναι μια κατασκευή ελέγχου μιας εισόδου και μιας εξόδου η οποία λαμβάνει τα δεδομένα εισόδου μέσω παραμέτρων και επιστρέφει το αποτέλεσμα με τον ίδιο τρόπο ή μέσω της δήλωσης επιστροφής.

Ο Wegner (1990) αναφέρει την βασική ορολογία, τις έννοιες και τους σκοπούς του **αντικειμενοστραφούς προγραμματισμού**. Σημαντικό ρόλο σε αυτό το υπόδειγμα προγραμματισμού έχουν τα αντικείμενα, τα οποία είναι συλλογές λειτουργιών που μοιράζονται μια κατάσταση. Κύριο συστατικό του αντικειμενοστραφούς προγραμματισμού είναι οι κλάσεις, οι οποίες χρησιμοποιούνται ως πρότυπα από τα οποία μπορούν να δημιουργηθούν αντικείμενα. Η έννοια της κληρονομικότητας, πρακτικά επιτρέπει την επαναχρησιμοποίηση μιας κλάσης για τον ορισμό νέων κλάσεων. Ένας από τους σκοπούς του αντικειμενοστραφούς υποδείγματος είναι η μείωση της πολυπλοκότητας και του χρόνου, μέσω της διάσπασης μιας μεγάλης εργασίας σε επιμέρους στοιχεία, την οποία πετυχαίνει με τη χρήση αντικειμένων, κλάσεων και κληρονομικότητας. Επίσης, επιτυγχάνεται η δημιουργία αντικειμενοστραφών βιβλιοθηκών και αποθετηρίων που επαναχρησιμοποιούνται μελλοντικά για την ανάπτυξη λογισμικού. Συμπερασματικά, ο αντικειμενοστραφής προγραμματισμός παρέχει ένα πλαίσιο για την διαχείριση των ογκωδών συστημάτων και προγραμμάτων.

Ο **συναρτησιακός προγραμματισμός** έχει ως βασική του λειτουργία την εφαρμογή συναρτήσεων στην είσοδό του (Hughes, 1990). Ένα συναρτησιακό πρόγραμμα αποτελεί μια συνάρτηση που λαμβάνει ως είσοδο του προγράμματος παραμέτρους και παραδίδει την έξοδο του προγράμματος ως αποτέλεσμα. Συνήθως, η κύρια συνάρτηση ορίζεται με την

περιγραφή άλλων συναρτήσεων, οι οποίες με τη σειρά τους ορίζονται με τον ορισμό ακόμη περισσότερων συναρτήσεων, έως το κατώτατο επίπεδο που οι συναρτήσεις πλέον είναι τα βασικά συστατικά της γλώσσας. Ουσιαστικά, στον συναρτησιακό προγραμματισμό γίνεται περιγραφή του «τι» ισχύει σε ένα πρόβλημα και όχι στο «πώς» επιλύεται αυτό (Σταματόπουλος, 2015).

Ο **προγραμματισμός βασισμένος σε γεγονότα** είναι ένα υπόδειγμα προγραμματισμού στο οποίο η ροή ελέγχου του προγράμματος καθορίζεται από μηνύματα, άλλα προγράμματα ή από ενέργειες του χρήστη (Jain & Nguyen, 2009). Σε τέτοιου είδους συστήματα δεν υφίσταται πλήρης καθορισμός της σειράς της εκτέλεσης από το ίδιο το πρόγραμμα, αλλά τα γεγονότα είναι αυτά που ελέγχουν τη ροή της εκτέλεσης. Τα γεγονότα μπορεί να πυροδοτήσουν άλλα γεγονότα, συναρτήσεις και λειτουργίες. Ουσιαστικά, τα γεγονότα αποτελούν σύνδεση μεταξύ αντικειμένων που στέλνουν μηνύματα. Το αντικείμενο λήψης, δηλαδή ο χειριστής γεγονότων, λαμβάνει το μήνυμα και εκτελεί τις ανάλογες εντολές. Ο χειριστής γεγονότων εξαρτάται από το περιεχόμενο του μηνύματος και από το πώς αυτό στάλθηκε. Μαζί με το μήνυμα μπορούν να αποσταλούν παράμετροι όπως άλλα γεγονότα ή συναρτήσεις και μεταβλητές με τις τιμές τους.

Ο Δημακόπουλος (2015) σημειώνει πως το υπόδειγμα του **παράλληλου προγραμματισμού** προτάθηκε για την επίλυση υπολογιστικών προβλημάτων που απαιτούν αυξημένη υπολογιστική ισχύ. Μέσω αυτού υπάρχουν δύο ή περισσότεροι επεξεργαστές που εκτελούν δύο ή περισσότερες εντολές ταυτόχρονα, σε αντίθεση με το σειριακό μοντέλο που εκτελείται μία εντολή τη φορά. Το πλήθος των επεξεργαστών που τρέχουν τα παράλληλα τμήματα κώδικα, μπορεί να αγγίζει αριθμούς που ξεπερνούν το εκατομμύριο. Τονίζεται πως η επικοινωνία μεταξύ αυτών των επεξεργαστών γίνεται είτε μέσω της κοινόχρηστης μνήμης είτε μέσω απευθείας συνδέσεων μεταξύ των επεξεργαστών. Για την αποτελεσματική εκτέλεση ενός παράλληλου προγράμματος τονίζεται η ανάγκη για συντονισμό και συνεργασία μεταξύ των επεξεργαστών που το εκτελούν. Επιπρόσθετα, η διαμοίραση των εργασιών που εκτελεί κάθε επεξεργαστής σε ένα υπολογιστικό πρόβλημα εξαρτάται από το ίδιο το πρόβλημα, από το πλήθος των επεξεργαστών, από τον τρόπο επικοινωνίας τους και από τις χρονικές καθυστερήσεις που αυτή συνεπάγεται.

1.3. Παράλληλος και ψευδοπαράλληλος προγραμματισμός

1.3.1. Παράλληλος προγραμματισμός

Πολλά επιστημονικά και μηχανικά προβλήματα απαιτούν μεγάλες ποσότητες επιμέρους υπολογισμών οι οποίοι πρέπει να πραγματοποιούνται σε φυσιολογικά χρονικά περιθώρια (Wilkinson & Allen, 2005). Προς την κατεύθυνση αυτή οι βελτιώσεις γίνονται κυρίως με την παραγωγή ταχυτέρων υλικών σε όλο και λιγότερο χώρο, κυρίως στον επεξεργαστή και την μνήμη μιας υπολογιστικής συσκευής (Δημακόπουλος, 2015). Το κυρίαρχο μοντέλο προγραμματισμού, που στοχεύουν οι παραπάνω βελτιώσεις, είναι το σειριακό κατά το οποίο ένας επεξεργαστής εκτελεί τις εντολές του κώδικα μία προς μία με τη σειρά που εμφανίζονται. Οι βελτιώσεις προς αυτή την κατεύθυνση έχουν φυσικά όρια που οδηγούν σε τεχνολογικά αδιέξοδα. Λύση στα παραπάνω όρια και αδιέξοδα προσπαθεί να δώσει ο παράλληλος προγραμματισμός που επίσης άρχισε να χρησιμοποιείται λόγω της ανάγκης για αύξηση της υπολογιστικής ισχύος των υπολογιστικών συστημάτων (Wilkinson & Allen, 2005).

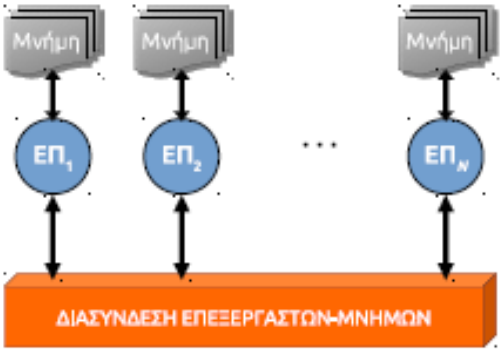
Εναλλακτικά, αντί της χρησιμοποίησης δύο ή περισσότερων επεξεργαστών στον παράλληλο προγραμματισμό, μπορεί πολλοί ηλεκτρονικοί υπολογιστές να δουλεύουν ταυτόχρονα πάνω στο ίδιο πρόβλημα τρέχοντας διαφορετικά τμήματα κώδικα του προβλήματος (Wilkinson & Allen, 2005). Ιδανικά, N (όπου N ένας φυσικός αριθμός) επεξεργαστές επιλύουν ένα πρόβλημα με N περισσότερες φορές υπολογιστική ισχύ από ότι ο ένας επεξεργαστής. Όμως, τα προβλήματα συχνά δεν μπορούν να διαιρεθούν τέλεια σε ανεξάρτητα τμήματα για λόγους ανάγκης πρόσβασης διαφορετικών τμημάτων στους ίδιους πόρους του προγράμματος. Επίσης, υπάρχουν προβλήματα συγχρονισμού και απαίτησης χρόνου για την επικοινωνία των επιμέρους τμημάτων τους (Δημακόπουλος, 2015).

Από τα διάφορα μοντέλα παράλληλου προγραμματισμού δύο είναι αυτά που κυρίως χρησιμοποιούνται περισσότερο (Kirk & Hwu, 2016). Αυτά είναι το μοντέλο Διασύνδεσης Μεταβίβασης Μηνυμάτων (Σχήμα 1.3.1) και το μοντέλο για συστήματα πολυεπεξεργαστών κοινόχρηστης μνήμης (Σχήμα 1.3.2). Στο Μοντέλο Διασύνδεσης Μεταβίβασης Μηνυμάτων δεν υπάρχει διαμοιρασμός της μνήμης μεταξύ των υπολογιστικών κόμβων. Η κοινή πρόσβαση σε πόρους, η κοινή χρήση δεδομένων και η μεταξύ τους αλληλεπίδραση πραγματοποιούνται αυστηρά μέσω ανταλλαγής μηνυμάτων. Το πεδίο που βρίσκει μεγάλη επιτυχία αυτό το μοντέλο παράλληλου προγραμματισμού είναι τα επιστημονικά

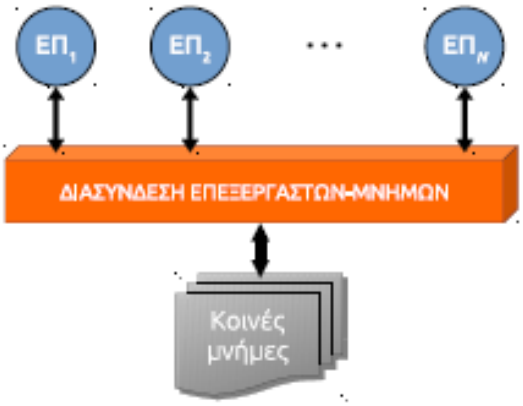
υπολογιστικά συστήματα υψηλής απόδοσης. Υπάρχουν περιπτώσεις υπολογιστικών συστημάτων που ακολουθούν αυτό το μοντέλο με πάνω από 100.000 κόμβους. Το μοντέλο αυτό θεωρείται δύσκολο, καθώς ο προγραμματιστής πρέπει να επιλέγει και να προγραμματίζει ο ίδιος πότε σταματάει η υπολογιστική διαδικασία και πραγματοποιούνται οι επικοινωνίες μέσω της μεταβίβασης μηνυμάτων (Δημακόπουλος, 2015). Το μοντέλο συστημάτων πολυεπεξεργαστών κοινόχρηστης μνήμης παρέχει στον προγραμματιστή κοινόχρηστη μνήμη στην οποία έχουν πρόσβαση όλοι οι κόμβοι του υπολογιστικού συστήματος (Kirk & Hwu, 2016). Σε αντίθεση με το προηγούμενο μοντέλο, αυτό δεν μπορεί να επεκταθεί σε πάνω από 200 κόμβους καθώς προκαλείται μεγάλη επιβάρυνση από την διαχείριση των νημάτων και των απαιτήσεων του υλικού.

Το επικρατέστερο μοντέλο από τα δύο παραπάνω είναι αυτό της κοινόχρηστης μνήμης (Δημακόπουλος, 2015). Από άποψη αρχιτεκτονικής και οργάνωσης η πιο οικονομική, απλή και συνηθισμένη μορφή είναι η ύπαρξη ενός διαύλου για τη σύνδεση και επικοινωνία μεταξύ των επεξεργαστών και των μνημών. Μέσα στον δίαυλο υπάρχουν καλώδια στα οποία «κυκλοφορούν» οι διευθύνσεις, τα δεδομένα και τα σήματα χρονισμού. Κάθε επεξεργαστής που είναι συνδεδεμένος στο δίαυλο έχει και μια δικιά του κρυφή μνήμη ώστε να μετριάζεται όσο είναι δυνατόν η προσπάθεια για επικοινωνία με την κύρια μνήμη που είναι και αυτή διασυνδεδεμένη στο δίαυλο. Η παρακολούθηση για το τι συμβαίνει στο δίαυλο είναι δυνατή από όλους άλλο μόνο ένα ζευγάρι τη φορά μπορεί να επικοινωνεί μέσω αυτού.

Η παραλληλία ενός προγράμματος μπορεί να χαρακτηριστεί με διαφόρους τρόπους (Bustard, 1990). Η μονάδα παραλληλίας είναι το γλωσσικό στοιχείο στο οποίο ορίζεται η συμπεριφορά της διεργασίας και συνηθώς αυτή είναι ένα τμήμα κώδικα. Ένα ακόμη στοιχείο που χαρακτηρίζει ένα παράλληλο πρόγραμμα είναι το επίπεδο παραλληλίας του, το οποίο είναι ο μέσος αριθμός των ενεργών διεργασιών κατά τη διάρκεια εκτέλεσης του προγράμματος. Άλλο στοιχείο που χαρακτηρίζει την παραλληλία ενός προγράμματος είναι η κλίμακα παραλληλίας. Αυτή είναι η μέση διάρκεια ζωής των διεργασιών κατά την εκτέλεση του προγράμματος. Τέλος, η παραλληλία χαρακτηρίζεται τον μέσο υπολογιστικό χρόνο ανάμεσα στις επικοινωνίες κατά τη εκτέλεση του προγράμματος.



Σχήμα 1.3. 1 Μοντέλο Διασύνδεσης Μεταβίβασης Κοινοχρήστων Μηνυμάτων (Δημακόπουλος, 2015)



Σχήμα 1.3. 2 Μοντέλο συστημάτων πολυεπεξεργαστών κοινόχρηστης μνήμης (Δημακόπουλος, 2015)

1.3.2 Προβλήματα παράλληλου προγραμματισμού και τρόποι επίλυσης

Για την συγγραφή κώδικα παράλληλων προγραμμάτων απαιτείται να αντιμετωπίζεται μια πληθώρα προβλημάτων (Bustard, 1990). Ένα τέτοιο πρόβλημα αποτελεί η παραβίαση του **αμοιβαίου αποκλεισμού**. Ορισμένες λειτουργίες σε ένα παράλληλο πρόγραμμα αποτυγχάνουν να παράγουν το επιθυμητό αποτέλεσμα όταν εκτελούνται όλες μαζί ταυτόχρονα. Τα τμήματα του κώδικα που αποτελούν τέτοιες λειτουργίες ονομάζονται **κρίσιμες περιοχές** και αν μια τέτοια περιοχή εκτελείται, πρέπει οι υπόλοιπες να εξαιρεθούν έως ότου τελειώσει. Το πρόβλημα αυτό δημιουργείται όταν ο προγραμματιστής δεν αναγνωρίζει τις κρίσιμες περιοχές κατά την συγγραφή του κώδικα. Ένα άλλο πρόβλημα είναι

τα **αδιέξοδα** που μπορεί να βρεθούν διαδικασίες ενός παράλληλου προγράμματος. Ένα αδιέξοδο μπορεί να συμβεί όταν οι παράλληλες διεργασίες προσπαθούν να επικοινωνήσουν, δηλαδή να αποστείλουν μηνύματα ταυτόχρονα και αποτυγχάνουν, είτε όταν δεν γίνεται αποτελεσματική διαχείριση των πόρων. Το αποτέλεσμα των αδιεξόδων είναι ο πρόωρος τερματισμός του προγράμματος. Πρόβλημα του παράλληλου προγραμματισμού επίσης αποτελούν οι **διαδικασίες που αναβάλλονται επ' αόριστο** καθώς περιμένουν ένα συμβάν που δεν πρόκειται να συμβεί. Αυτή η κατάσταση μπορεί να προκύψει όταν τα αιτήματα για πόρους διαχειρίζονται από αλγορίθμους που δεν δίνουν προτεραιότητα με βάση το χρόνο αναμονής. Για την αποφυγή αυτού του προβλήματος υπάρχουν τεχνικές που προτείνουν οι ανταγωνιζόμενες διεργασίες να παίρνουν υψηλότερη προτεραιότητα , με κριτήριο το πόσο έχουν περιμένει. Επίσης, σε ένα παράλληλο πρόγραμμα ζήτημα αποτελεί και η **δίκαια πρόοδος όλων των διεργασιών**. Όταν μερικές διεργασίες παραμελούνται αρκετά όσον αφορά την παροχή πόρων μπορεί να οδηγήσει το πρόγραμμα σε λάθη. Επιπρόσθετα , ένα ζήτημα που δεν αποτελεί λάθος αλλά οδηγεί στην σπατάλη της υπολογιστικής ισχύος ενός επεξεργαστή είναι αυτό της **απασχολημένης αναμονής**. Χαρακτηριστικό παράδειγμα αυτού του φαινομένου είναι όταν ένας επεξεργαστής αναμένει να ικανοποιηθεί κάποια συνθήκη για να τρέξει ένα τμήμα κώδικα. Ως ότου συμβεί αυτό, η μόνη λειτουργία που εκτελεί είναι ο έλεγχος της συνθήκη ξανά και ξανά. Τέλος, εμφανίζονται τα παροδικά σφάλματα, τα οποία μπορεί να συμβούν αλλά μπορεί και όχι ανάλογα με το μονοπάτι εκτέλεσης που προκύπτει από την εκάστοτε εκκίνηση του προγράμματος. Αυτά τα λάθη πολλές φορές είναι δύσκολο να εντοπιστούν ή να εντοπιστεί ο λόγος εμφάνισης τους και γενικά διακρίνονται μέσω του πειραματισμού.

1.3.3. Ψευδοπαράλληλος προγραμματισμός

Στο εύρος του παράλληλου προγραμματισμού υπάρχει και μια άλλη έννοια, αυτή του ψευδοπαράλληλου προγραμματισμού (Bal et al., 1989). Ένας επεξεργαστής με μεγάλη ταχύτητα μπορεί να εκτελεί πολλαπλές διεργασίες ταυτόχρονα εναλλάσσοντας ταχύτατα την λειτουργία του από την εκτέλεση μιας διεργασίας στην άλλη και ξανά το ίδιο. Με αυτόν τον τρόπο δημιουργείται η ψευδαίσθηση ότι πολλές διεργασίες εκτελούνται πραγματικά ταυτόχρονα. Αυτή η εναλλαγή διεργασιών κατά την εκτέλεση από έναν επεξεργαστή ονομάζεται ψευδοπαραλληλισμός. Σε πολλές γλώσσες προγραμματισμού δεν γίνεται εμφανές στον προγραμματιστή εάν προγραμματίζει παράλληλα ή ψευδοπαράλληλα. Όπως

και στον παράλληλο προγραμματισμό, η χρήση του ψευδοπαράλληλου προγραμματισμού συνεισφέρει στην εξοικονόμηση χρόνου, όμως τα προβλήματα συγχρονισμού και επικοινωνίας παραμένουν.

1.4. Διδακτική και διδασκαλία του προγραμματισμού

Οι ηλεκτρονικοί υπολογιστές είναι η επικρατούσα τεχνολογία του εικοστού πρώτου αιώνα. Ο προγραμματισμός τους, δηλαδή η κατασκευή λογισμικού αποτελεί μια θεμελιώδη δραστηριότητα και κατ' επέκταση διδάσκεται παγκοσμίως σε όλες τις βαθμίδες της εκπαίδευσης (Michaelson, 2015). Εδώ και εβδομήντα χρόνια έχουν αναπτυχθεί αρκετές προσεγγίσεις για τη διδακτική του προγραμματισμού. Η διδασκαλία του προγραμματισμού συνδέεται με την διδασκαλία των τεχνικών για την επίλυση προβλημάτων. Επίσης, ο προγραμματισμός με την επίλυση προβλημάτων συνδέονται μέσω της υπολογιστικής σκέψης, όπως την περιέγραψε η Wing (2006).

Η διδακτική του προγραμματισμού σε αρχικά μαθήματα συνήθως γίνεται μέσω της παραδοσιακής διδακτικής προσέγγισης κατά την οποία **ο εκπαιδευτικός παρουσιάζει τις έννοιες και τις δομές της γλώσσας προγραμματισμού που διδάσκει** (Lidtke & Zhou, 1999). Η παραδοσιακή διδασκαλία προγραμματισμού πραγματοποιείται με την παρουσίαση του συντακτικού της προγραμματιστικής γλώσσας, τις δομές ακολουθίας, επιλογής και επανάληψης, τους αφηρημένους τύπους δεδομένων, τις δομές δεδομένων, τις συναρτήσεις και την πολυπλοκότητα των αλγορίθμων. Όμως, με αυτόν τον τρόπο, οι μαθητές σκέφτονται σε ένα στενό πλαίσιο και λανθασμένα καταλήγουν στο συμπέρασμα ότι το δύσκολο και το πιο σημαντικό κομμάτι του προγραμματισμού είναι η μετατροπή της επίλυσης ενός προβλήματος σε μια συγκεκριμένη γλώσσα προγραμματισμού και όχι η ίδια η επίλυση του προβλήματος. Οι μαθητές δουλεύοντας απομονωμένα ο ένας από τον άλλον έρχονται σε σύγχυση καθώς δεν γνωρίζουν αν τα λάθη τους προέρχονται από ζητήματα αδυναμίας στην συγκεκριμένη προγραμματιστική γλώσσα που δουλεύουν ή σε ζητήματα αδυναμίας κατανόησης του προβλήματος και δημιουργίας αποτελεσματικού αλγορίθμου.

Ένα άλλο μοντέλο διδασκαλίας του προγραμματισμού είναι αυτό που οι μαθητές δουλεύουν σε δυάδες, το οποίο παραπέμπει στην ομαδική ή και **συνεργατική μάθηση** (Williams & Upchurch, 2001). Σε αυτό το μοντέλο διδασκαλίας οι δύο εκπαιδευόμενοι προγραμματιστές, εργάζονται συνεργατικά πάνω στον ίδιο αλγόριθμο σχεδιάζοντας και υλοποιώντας τον

κώδικα. Υπάρχουν πολλοί τρόποι εφαρμογής του μοντέλου αυτού, με έναν από αυτούς να προτείνει να δίνεται ο έλεγχος του μολυβιού και του χαρτιού ή του πληκτρολογίου και του ποντικιού στον έναν εκπαιδευόμενο, ενώ ο άλλος παρατηρεί, ελέγχει, σκέφτεται εναλλακτικές προτάσεις και στρατηγικές και προσπαθεί να εντοπίζει τα προβλήματα. Τα αποτελέσματα αυτής της προσέγγισης δείχνουν μια σημαντική μείωση στο χρόνο που απαιτείται για την υλοποίηση των αλγορίθμων από τους μαθητές αλλά και μια σημαντική μείωση στο φόρτο εργασίας που αναλαμβάνει κάθε εκπαιδευόμενος. Επίσης, οι μαθητές δουλεύοντας σε δυάδες απολαμβάνουν περισσότερο τη διαδικασία από το να δουλεύουν μόνοι τους. Η συνεχής παρατήρηση του κώδικα από τους συνεργάτες οδηγεί στην επίλυση λαθών και στην εύρεση βελτιστοποιημένων λύσεων, με τους ίδιους τους εκπαιδευόμενους να τονίζουν πως ο ένας βοηθούσε τον άλλον στη διαδικασία της μάθησης αλλά και πως η επίλυση προβλημάτων έγινε πραγματικότητα σε μικρότερους χρόνους μέσω της συνεργασίας. Επιπρόσθετα, μέσω της συνεργασίας αναπτύσσονται οι δεξιότητες αποτελεσματικής επικοινωνίας μέσα στην ομάδα.

Μια διαφορετική τεχνική διδασκαλίας νέων εννοιών προγραμματισμού είναι αυτή του “**μαύρου κουτιού**” (Haberman & Kolikant, 2001). Στην προσέγγιση αυτή οι μαθητές έρχονται σε επαφή με τις προγραμματιστικές έννοιες καθώς πραγματοποιούν διάφορες δραστηριότητες και ύστερα συμμετέχουν σε μια συζήτηση στην τάξη που αφορά τις έννοιες που διδάχθηκαν. Κατά την πρώτη δραστηριότητα, οι μαθητές εκτελούν ένα πρόγραμμα στον υπολογιστή, στον κώδικα του οποίου δεν έχουν πρόσβαση. Αλληλοεπιδρούν με το πρόγραμμα μέσω των εισόδων που ζητάει και μέσω των εξόδων που δίνει. Κατά τη δεύτερη δραστηριότητα, δίνεται ο κώδικας στους μαθητές και αυτοί αναγνωρίζουν τα τμήματα κώδικα που προκάλεσαν την αλληλεπίδραση μαζί τους. Στην συζήτηση με όλη την τάξη που ακολουθεί ο εκπαιδευτικός δίνει απαντήσεις σε τυχόν ερωτήματα και προβληματισμούς που προέκυψαν. Στόχος της προσέγγισης είναι η εισαγωγή των μαθητών στις έννοιες μέσω της ενεργητικής συμμετοχής τους. Τα αποτελέσματα της εφαρμογής του “μαύρου κουτιού” δείχνουν μια καλύτερη αφομοίωση των διδασκόμενων εννοιών σε σχέση με την παραδοσιακή διδασκαλία.

Από τον Lischner (2001) προτάθηκε η διδασκαλία του προγραμματισμού μέσω **διερεύνησης** η οποία στηρίζεται στην θεωρία του εποικοδομητισμού. Η εφαρμογή του διερευνητικού μοντέλου διδασκαλίας γίνεται συνήθως σε εργαστηριακές δραστηριότητες κατά τις οποίες

οι μαθητές καλούνται να διαβάσουν ένα μικρό πρόγραμμα που τους δίνεται πρόσβαση. Ύστερα, απαντούν σε ερωτήσεις πάνω στις λειτουργίες του προγράμματος και στα αποτελέσματα των προγραμματιστικών δομών του. Ουσιαστικά, καλούνται να προβλέψουν τις εξόδους και τη συμπεριφορά του προγράμματος. Μετά από αυτή την φάση, γίνεται σύγκριση των προβλέψεων των μαθητών σε σχέση με τα πραγματικά αποτελέσματα. Στις περιπτώσεις που οι απαντήσεις είναι διαφορετικές από τα αποτελέσματα, οι εκπαιδευόμενοι εξηγούν τις επιλογές τους και κατευθύνονται ώστε να εντοπίσουν τους λόγους των λανθασμένων ερμηνειών τους. Εάν δεν το καταφέρουν από μόνοι τους, ο διδάσκων παρεμβαίνει και τους παρέχει βοήθεια ώστε να τα αναγνωρίσουν. Τα αποτελέσματα της διερευνητικής διδασκαλίας στον προγραμματισμό μετριούνται μέσω τεστ πριν και μετά την πραγματοποίηση της δραστηριότητας και δείχνουν να έχουν θετικό πρόσημο για το πόσο αποτελεσματική είναι η προσέγγιση.

Από πολλές μελέτες προκύπτει ότι οι αρχάριοι προγραμματιστές αντιμετωπίζουν σημαντικές δυσκολίες (Robins, Rountree & Rountree, 2003). Σε αντίθεση με τους έμπειρους προγραμματιστές, δεν αφιερώνουν τον απαραίτητο χρόνο για το κατάλληλο σχεδιασμό και για τον έλεγχο του κώδικα τους. Επίσης, προτιμούν να διορθώνουν τον κώδικά τους τοπικά και όχι να αναδημιουργούν μεγαλύτερα τμήματα του, καθώς δίνουν σημασία σε κάθε σειρά ξεχωριστά και όχι σε μια ολοκληρωμένη δομή ή ένα μεγάλο τμήμα κώδικα. Επιπλέον, προβλήματα αντιμετωπίζουν και στην κατανόηση της συνέχειας του κώδικα και στο να αντιληφθούν πως η εκτέλεση κάθε εντολής είναι άμεσα συνδεδεμένη με την εκτέλεση των προηγουμένων εντολών. Πολλές φορές από τους αρχάριους προγραμματιστές λείπουν τα κατάλληλα νοητικά μοντέλα, αποτυγχάνουν να εφαρμόσουν τη σχετική γνώση και χρησιμοποιούν γενικές στρατηγικές επίλυσης προβλημάτων. Όλα τα παραπάνω δείχνουν πως η διδασκαλία του προγραμματισμού σε αρχάριους δεν είναι μια εύκολη και απλή διαδικασία (Lahtinen, Ala-Mutka & Järvinen, 2005). Μάλιστα, έχουν παρατηρηθεί μεγάλα ποσοστά μαθητών που εγκαταλείπουν την προσπάθεια ανάπτυξης προγραμματιστικών ικανοτήτων.

1.5. Διδασκαλία του προγραμματισμού στο δημοτικό με Scratch

1.5.1. Διδασκαλία του προγραμματισμού στο δημοτικό σχολείο

Η διδασκαλία προγραμματισμού σε παιδιά μικρών ηλικιών ξεκινάει ήδη από την δεκαετία του 1960 με την προγραμματιστική γλώσσα Logo που δημιουργήθηκε στα τέλη της δεκαετίας αυτής. Αυτή η προγραμματιστική γλώσσα σχεδιάστηκε ώστε να παρέχει μια εννοιολογική βάση για τη διδασκαλία μαθηματικών και λογικών τρόπων σκέψης στα πλαίσια του προγραμματισμού. Πέραν από την χρησιμοποίηση της Logo για τους νεαρούς μαθητές, δεν χρησιμοποιήθηκαν πολλές προγραμματιστικές γλώσσες έως ότου εμφανιστούν σε αυτό το χώρο οι γλώσσες οπτικού προγραμματισμού (Kalelioğlu, 2015).

Η Logo και οι άλλες γλώσσες που χρησιμοποιήθηκαν αρχικά για τη διδασκαλία του προγραμματισμού σε παιδιά είχαν προβλήματα τα οποία δεν έγινε εφικτό να ξεπεραστούν (Resnick et al., 2009). Οι πρώτες αυτές γλώσσες ήταν αρκετά δύσκολες στη χρήση τους από νεαρούς μαθητές και έτσι και πολλοί από αυτούς δεν κατάφεραν να κατακτήσουν το συντακτικό τους. Επιπλέον, ο προγραμματισμός συχνά γινόταν με αλγόριθμους όπως η εύρεση των πρώτων αριθμών, οι οποίοι δεν βρίσκονται στα ενδιαφέροντα των μαθητών. Τέλος, οι προγραμματιστικές έννοιες εισηγούνταν χωρίς να υπάρχει καθοδήγηση όταν υπήρχαν δυσκολίες, και χωρίς ενθάρρυνση για περαιτέρω εμβάθυνση όταν τα πράγματα πήγαιναν καλά.

Για να αναπτυχθεί ένα πρόγραμμα απαιτείται η ύπαρξη προγραμματιστικής και αλγοριθμικής λογικής. Όσον αφορά τις νεαρές ηλικίες, οι μαθητές δείχνουν να αισθάνονται πιο άνετα και να επιτυγχάνουν καλύτερα αποτελέσματα, μαθαίνοντας προγραμματισμό μέσω οπτικών παρουσιάσεων, λεκτικών εξηγήσεων και ανακαλύπτοντας στοιχεία μόνοι τους (Zhang et al., 2014). Μέσω των περιβαλλόντων οπτικού προγραμματισμού οι νεαροί μαθητές εξοικειώνονται με την επιστήμη των ηλεκτρονικών υπολογιστών και με τις έννοιες της πληροφορικής αλλά και αναβαθμίζουν τις δεξιότητες σκέψης υψηλού επιπέδου. Επίσης, οι στρατηγικές εκμάθησης μέσω δοκιμής-λάθους αυξάνουν την δεξιότητες των νεαρών μαθητών για την επίλυση υπολογιστικών προβλημάτων (Liu, Cheng & Huang, 2011). Ακόμη, οι νεαροί μαθητές που ασχολήθηκαν με τον προγραμματισμό ρομποτικής βελτίωσαν τις δεξιότητές τους όσον αφορά τη γεωμετρική τους αντίληψη καθώς προγραμμάτιζαν (Keren & Fridin, 2014).

Οι Rich et al. (2019) σε μελέτη τους, που περιλαμβάνει πολλές χώρες και συμμετέχοντες, ερευνούν τις διεθνείς τάσεις στην διδασκαλία του προγραμματισμού στο δημοτικό και στις πρώτες τάξεις του γυμνασίου. Καταλήγουν στο ότι, δύο στους τρεις καθηγητές έχουν δηλώσει θετική διάθεση για να τον διδάξουν. Το μεγαλύτερο θέμα που έχουν οι εκπαιδευτικοί με τον προγραμματισμό φαίνεται να είναι η αυτοπεποίθησή τους στην προγραμματιστική και υπολογιστική τους ικανότητα, χωρίς αυτό όμως να αποτελεί τροχοπέδη και τελικά να διδάσκουν προγραμματισμό σε νεαρούς μαθητές. Οι μαθητές στην πλειοψηφία τους δείχνουν να απολαμβάνουν το μάθημα του προγραμματισμού, τάση η οποία ενισχύει την κατεύθυνση θετικής στάσης που ήδη έχουν οι εκπαιδευτικοί. Επίσης, για τη διδασκαλία προγραμματισμού σε αρχάριους είναι σημαντική η παροχή βοήθειας από τους εκπαιδευτικούς όταν οι μαθητές δεν μπορούν να προχωρήσουν.

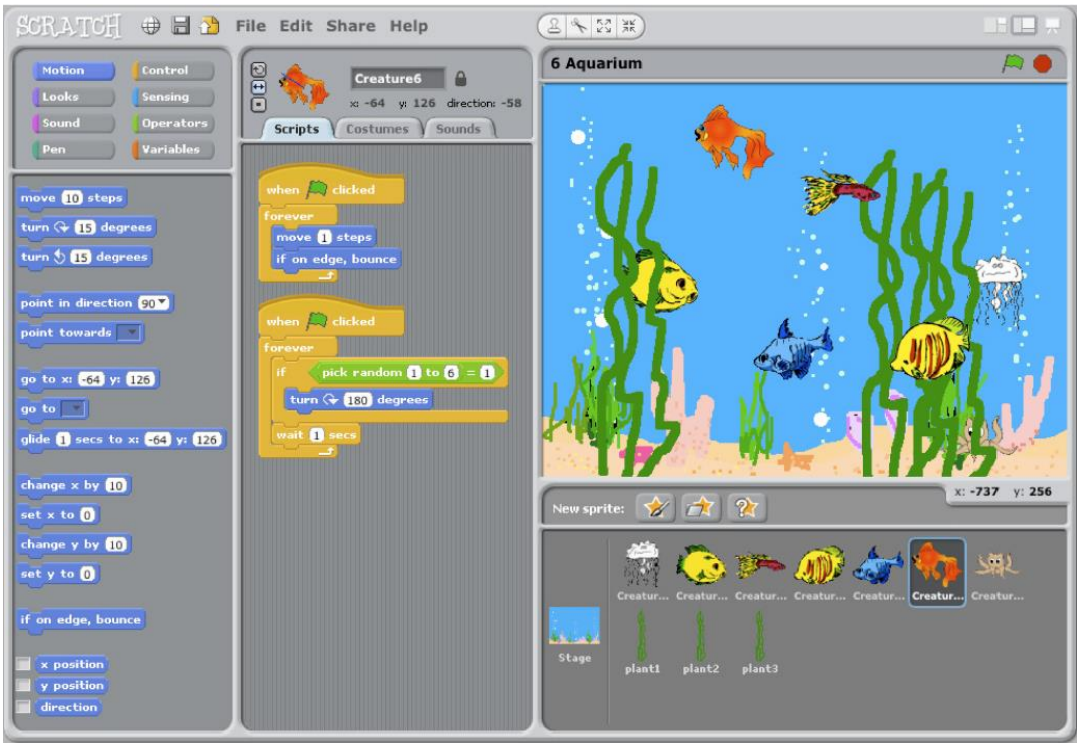
Οι γλώσσες που χρησιμοποιούνται κατά κόρον για τη διδασκαλία προγραμματισμού σε νεαρές ηλικίες είναι αυτές του οπτικού προγραμματισμού (Sáez-López, Román-González & Vázquez-Cano, 2016). Η χρήση αυτού του τύπου γλωσσών προγραμματισμού βελτιώνει διάφορα στοιχεία του εκπαιδευόμενου όπως την εκμάθηση εννοιών προγραμματισμού, τη λογική και την υπολογιστική σκέψη. Η εκμάθηση μέσω των γλωσσών αυτών παρέχει στους μαθητές διασκέδαση, ενθουσιασμό, κίνητρο και αφοσίωση στο μάθημα. Οι εκπαιδευόμενοι επιτυγχάνουν υψηλό βαθμό κατανόησης των δομών ακολουθίας, επιλογής, επανάληψης, του παραλληλισμού και στη κοινοχρησίας περιεχομένου. Επίσης, η χρήση τέτοιου τύπου γλωσσών ενισχύει την διδασκαλία μέσω ενεργητικής μάθησης.

1.5.2. Scratch

Μία από τις περισσότερο χρησιμοποιούμενες πλατφόρμες οπτικού προγραμματισμού είναι το Scratch και αυτό γιατί ξεπέρασε σε μεγάλο βαθμό τα προβλήματα των πρώιμων προγραμματιστικών γλωσσών που προορίζονταν για την εκπαίδευση νεαρών μαθητών (Resnick et al., 2009). Το Scratch είναι ένα περιβάλλον οπτικού προγραμματισμού, που επιτρέπει στους χρήστες να δουλεύουν και να δημιουργούν ψηφιακά παιχνίδια, εικονογραφημένες ιστορίες και προσομοιώσεις προγραμματίζοντας (Maloney, et al., 2010). Η δημιουργία γίνεται με την χρήση προγραμματιστικών σεναρίων και πολυμεσικού υλικού. Το συγκεκριμένο περιβάλλον οπτικού προγραμματισμού δίνει τη δυνατότητα για διαχείριση πολυμεσικού υλικού, το οποίο είναι δημοφιλές στις νεαρές ηλικίες, και ευνοεί τη μάθηση μέσω διερεύνησης αντί για τη μάθηση μέσω αυστηρών οδηγιών όπως γίνεται με άλλα

περιβάλλοντα προγραμματισμού. Η ανάπτυξη της συγκεκριμένης πλατφόρμας έγινε και με το κριτήριο της προσφοράς νοήματος στο χρήστη (Resnick et al., 2009). Πιο συγκεκριμένα, υπάρχει η δυνατότητα για τη δημιουργία πληθώρας διαφορετικών εργασιών με ποικίλα χαρακτηριστικά αλλά και για εξατομίκευση των εργασιών του χρήστη καθώς έχει την επιλογή να εισάγει προσωπικές του φωτογραφίες, βίντεο και ήχους. Η βάση στην οποία δημιουργήθηκε αυτή η πλατφόρμα είναι οι κονστρουκτιβιστικές ιδέες της Logo και η απλοποίηση της εισαγωγής και δημιουργίας πολυμέσων (Maloney et al., 2010). Το Scratch έχει ως στόχο να εισάγει προγραμματιστικές έννοιες σε εκπαιδευόμενους χωρίς προηγούμενη εμπειρία, πράγμα το οποίο ευνοεί τη διδασκαλία του προγραμματισμού στο δημοτικό. Προς αυτήν την κατεύθυνση κινείται η ύπαρξη οπτικών πλακιδίων, η δομή της οθόνης διάδρασης με το χρήστη και το μικρό σε αριθμό πακέτο διαθέσιμων εντολών.

Η οθόνη αλληλεπίδρασης του Scratch επιτρέπει την εύκολη πλοήγηση του χρήστη (Maloney et al., 2010). Όλα τα σημαντικά στοιχεία που απαρτίζουν το Scratch βρίσκονται ταυτόχρονα στην οθόνη και είναι πάντα ορατά από το χρήστη. Η οθόνη του Scratch αποτελείται από τέσσερα τμήματα (Σχήμα 1.5.1). Το αριστερό τμήμα της οθόνης αποτελείται από τις κατηγορίες των εντολών σε μορφή πλακιδίων-μπλοκ, που με κάθε επιλογή κατηγορίας εμφανίζεται από κάτω το σύνολο των εντολών που ανήκουν σε αυτή. Οι κατηγορίες αυτές είναι οχτώ: κίνηση, έλεγχος, όψεις, αισθητήρες, ήχος, συμβάντα, τελεστές και μεταβλητές. Το μεσαίο υποπαράθυρο απαρτίζεται από ένα επιλεγόμενο στοιχείο με καρτέλες για τον κώδικα, τις εικόνες και τον ήχο του. Πάνω δεξιά βρίσκεται ένα τμήμα του παραθύρου η οποία αποτελεί την σκηνή στην οποία λαμβάνει χώρα η δράση των στοιχείων που βασίζεται στον κώδικα. Τέλος, κάτω δεξιά βρίσκονται οι μικρογραφίες όλων των υπαρχουσών στοιχείων, με υπογραμμισμένο το τρέχων επιλεγόμενο.



Σχήμα 1.5. 1 Η οθόνη του Scratch (Maloney et al., 2010)

Το Scratch χαρακτηρίζεται από ζωτικότητα, ευελιξία και κοινωνικότητα (Maloney et al., 2010). Με την έννοια της ζωτικότητας νοείται πως η διαδικασία συγγραφής του προγράμματος δεν χωρίζεται στη συγγραφή του κώδικα και στην συνέχεια στην εκτέλεσή του. Αυτές οι δύο λειτουργίες μπορούν να συμβούν παράλληλα χωρίς να διαχωρίζονται από το βήμα της μεταγλώττισης του κώδικα. Ο χρήστης μπορεί να αλλάζει τα μπλοκ του κώδικα και διάφορες παραμέτρους καθώς ο κώδικας εκτελείται. Επιπρόσθετα, με ένα απλό κλικ σε ένα μπλοκ άμεσα αρχίζουν να εκτελούνται τμήματα κώδικα που το απαρτίζουν (Resnick et al., 2009). Μέσω της ζωτικότητας οι αρχάριοι χρήστες μπορούν να κάνουν ευκολότερα έλεγχο και διορθώσεις του κώδικα που αποσκοπούν στη βελτίωσή του (Maloney et al., 2010). Η έννοια της ευελιξίας προσεγγίζει τη δυνατότητα που παρέχει το Scratch στους χρήστες να πειραματίζονται με τις εντολές και τα τμήματα κώδικα. Επίσης, η περιοχή για συγγραφή κώδικα στο Scratch είναι σαν μια επιφάνεια εργασίας (Resnick et al., 2009). Η ευελιξία αυτή επιτρέπει μια μορφή πρακτικής μάθησης καθώς ο χρήστης πειραματίζεται ο ίδιος και ανακαλύπτει την λειτουργία των μπλοκ. Για την περαιτέρω βοήθεια των χρηστών οι μεταβλητές έχουν προκαθορισμένες τιμές στα μπλοκ ώστε να μπορεί να εκτελεστεί άμεσα ο κώδικας με τη δημιουργία του. Επίσης, δεν χρειάζεται να ολοκληρωθεί ο κώδικας για να

τρέξει, πράγμα το οποίο ενισχύει την ευελιξία του (Maloney et al., 2010). Η πλατφόρμα επιτρέπει το διαμοιρασμό των εργασιών ενός χρήστη στο διαδίκτυο ώστε να τρέξει τον κώδικα αυτόν ένα άλλο άτομο και να υπάρξει αλληλεπίδραση μεταξύ τους και πιθανός σχολιασμός. Η παραπάνω διαδικασία προσφέρει κοινωνικότητα στους χρήστες.

Όταν τρέχει ένα τμήμα κώδικα, το μπλοκ στο οποίο βρίσκεται φωτίζεται, ώστε να δείχνει στον χρήστη τι ακριβώς εκτελείται, σε περίπτωση που υπάρχει σύγχυση (Maloney et al., 2010). Επιπρόσθετα, δεν υπάρχουν μηνύματα λάθους. Όταν υπάρχουν λάθη, το Scratch είναι κατασκευασμένο έτσι ώστε να προσπαθεί να κάνει από μόνο του κάτι που βγάζει νόημα. Τα λάθη φυσικά δεν εξαφανίζονται και ο προγραμματιστής οφείλει να είναι προσεκτικός. Επίσης, οι μεταβλητές είναι ορατές σε αντίθεση με τις γλώσσες προγραμματισμού που δεν έχουν οπτικό περιβάλλον, στις οποίες οι μεταβλητές είναι αόρατες. Ο χρήστης βλέποντας τις μεταβλητές να αλλάζουν σχηματίζει μια καλύτερη εικόνα για το τι είναι μια μεταβλητή και πώς συμπεριφέρεται. Επιπλέον, η οπτική τους απόδοση μπορεί να χρησιμεύσει σαν ταμπλό σκορ σε ένα παιχνίδι ή κάτι παρεμφερές. Το σύνολο των διαθέσιμων εντολών είναι μικρό, γεγονός το οποίο συμβαίνει λόγω του ότι ένα μεγάλο πλήθος διαθέσιμων εντολών θα κατελάμβανε πολύ χώρο στην οθόνη. Καθώς η πλατφόρμα εξελίσσεται, ο αριθμός των εντολών έχει παραμείνει μικρός και έχει χωριστεί σε κατηγορίες που εμφανίζονται με σε ένα κυλιόμενο μενού.

Το Scratch είναι μια πλατφόρμα που έχει ως στόχο την εισαγωγή του χρήστη στον κόσμο του προγραμματισμού μέσω των ενδιαφερόντων του (Resnick et al., 2009 · Maloney et al., 2010). Η οθόνη εργασίας, ο μικρός αριθμός εντολών, η ζωτικότητα, η ευελιξία, η εξάλειψη λαθών, η κοινωνικότητα, η έλλειψη του βήματος μεταγλώττισης και η αποφυγή εμφάνισης λαθών στοχεύουν στην απλότητα της πλατφόρμας και στην εκμάθηση του προγραμματισμού μέσω διερεύνησης από τους εκπαιδευόμενους (Resnick et al., 2009· Maloney et al., 2010).

1.6. Βιβλιογραφική ανασκόπηση παράλληλου προγραμματισμού στο δημοτικό και τις πρώτες τάξεις γυμνασίου

1.6.1. Μεθοδολογία

Η αναζήτηση επιστημονικών άρθρων έγινε στις ακαδημαϊκές βάσεις δεδομένων Scopus, ACM, ScienceDirect, IEEE και Google Scholar με χρονικό περιορισμό την ημερομηνία

δημοσίευσης το 2010 και μετά. Οι λέξεις και οι φράσεις κλειδιά χρησιμοποιήθηκαν στον αλγόριθμο αναζήτησης με τους παρακάτω συνδυασμούς: (("parallel programming" OR "parallel computing" OR "concurrent computing" OR "concurrent programming" OR "parallel outcomes") AND (education OR teaching OR lesson) AND (primary OR elementary OR secondary OR children OR young OR beginners)). Οι παραπάνω συνδυασμοί προέκυψαν καθώς στην αγγλική βιβλιογραφία ο παράλληλος προγραμματισμός συναντάται και ως ταυτόχρονος προγραμματισμός αλλά και ως παράλληλη ή ταυτόχρονη υπολογιστική. Επίσης, οι νεαρές ηλικίες δημοτικού ανταποκρίνονται στις λέξεις primary, elementary, children, young, beginners, και οι πρώτες τάξεις γυμνασίου με τη λέξη secondary.

Ο αλγόριθμος αυτός επιστρέφει 8 άρθρα. Πιο συγκεκριμένα, στη βάση δεδομένων του Scopus επιστρέφει 38 αποτελέσματα εκ των οποίων τα 7 αφορούν παράλληλο προγραμματισμό και εκπαίδευση σε νεαρές ηλικίες και είναι στα αγγλικά. Από τα υπόλοιπα 31, υπάρχουν 3 τα οποία φαίνεται από την περίληψή τους, η οποία είναι γραμμένη στην αγγλική γλώσσα, να ανταποκρίνονται στο θέμα της διδασκαλίας του παράλληλου προγραμματισμού σε νεαρούς μαθητές. Όμως, το σώμα του υπόλοιπου κειμένου τους είναι γραμμένο στην ρωσική και την ισπανική γλώσσα οπότε δεν χρησιμοποιήθηκαν στην παρούσα βιβλιογραφική ανασκόπηση. Τα υπόλοιπα 28 αφορούν κυρίως την εκπαίδευση φοιτητών πληροφορικής. Ο αλγόριθμος στην ηλεκτρονική βιβλιοθήκη της ACM ελέγχοντας τίτλους, λέξεις κλειδιά και περιλήψεις των άρθρων για το 2010 και μετά, επέστρεψε 20 αποτελέσματα εκ των οποίων τα 2 αφορούν παράλληλο προγραμματισμό, εκπαίδευση σε νεαρές ηλικίες αλλά και τα 2 τα εντοπίζει η αναζήτηση στο Scopus. Στις βάσεις δεδομένων ScienceDirect και IEEE επιστρέφονται 45 και 106 άρθρα αντίστοιχα χωρίς κάποιο από αυτά να αφορά παράλληλο προγραμματισμό και νεαρές ηλικίες. Τέλος, το Google Scholar επιστρέφει πληθώρα άρθρων με 3 από αυτά να είναι σχετικά με παράλληλο προγραμματισμό και νεαρές ηλικίες, αλλά μόνο το ένα να μην έχει ήδη βρεθεί στη Scopus.

1.6.2. Βιβλιογραφική ανασκόπηση

Ο Tarkan et al. το 2010 δημιούργησαν με την συμμετοχή εννέα μαθητών δημοτικού, ηλικίας 7 έως 11 ετών, μια απλή γλώσσα οπτικού προγραμματισμού που αφορά την μαγειρική, την Toque. Σκοπός της γλώσσας αυτής είναι η χρησιμοποίησή της για τη διδασκαλία της υπολογιστικής σκέψης στους μαθητές του δημοτικού. Τα προγράμματα που προκύπτουν από αυτή τη γλώσσα είναι εικονογραφημένες συνταγές μαγειρικής που ένας εικονικός σεφ

προετοιμάζει εικονικά πιάτα σε ένα γραφικό περιβάλλον κουζίνας. Αυτές οι εμπειρίες σχεδίασης, εκτός τον άλλων, έδειξαν και κάποια αποτελέσματα που αφορούν τον παράλληλο προγραμματισμό. Οι μαθητές εξέφρασαν ως επιθυμία να υπάρχουν αρκετοί σεφ οι οποίοι παράλληλα ετοιμάζουν διαφορετικά πιάτα με διαφορετικές συνταγές αντί για έναν σεφ που θα κάνει αρκετές επαναλήψεις, κάτι το οποίο παραπέμπει στον παράλληλο προγραμματισμό. Πιο συγκεκριμένα, χαρακτήρισαν την επανάληψη ως βαρετή και ως απάντηση στο πώς να γίνει πιο ενδιαφέρουσα η γλώσσα, έδωσαν την αύξηση των σεφ που δουλεύουν παράλληλα. Σύμφωνα με τα παραπάνω, ο Tarkan et al., αναφέρουν πως υπάρχει πιθανότητα οι μαθητές να προτιμούν την παράλληλη επεξεργασία από τη δομή επανάληψης.

Το 2012 οι Gregg, Tychonievich, Cohoon και Hazelwood δημοσίευσαν ένα άρθρο που αφορά τη διδασκαλία παράλληλου προγραμματισμού σε μαθητές ηλικίας 9 και 10 ετών μέσω της γλώσσας EcoSim η οποία επιτρέπει την παράλληλη επεξεργασία. Η EcoSim χαρακτηρίζεται από εμφανή παραλληλία, αφού οι πολλαπλοί επεξεργαστές γίνονται φανεροί στο χρήστη, αλλά και από ευκολία στην κατανόηση για να μην υπάρχει σύγχυση στη χρήση της. Επίσης, είναι βασισμένη στο διαδίκτυο ώστε οι μαθητές να δουλεύουν από το σπίτι. Η EcoSim είναι κατασκευασμένη ώστε να κινεί το ενδιαφέρον των μαθητών μέσω του γραφικού περιβάλλοντός της. Οι μαθητές που συμμετείχαν στην έρευνα ήταν αρχάριοι στον προγραμματισμό χωρίς προηγούμενη εμπειρία. Επιπλέον, τα μαθήματα έγιναν με στόχο την εισαγωγή των βασικών ιδεών του παράλληλου προγραμματισμού στους μαθητές και την απόκτηση της δυνατότητας να σκέφτονται παράλληλα για την επίλυση προβλημάτων, μέσω της χρήσης πολλαπλών επεξεργαστών. Στη συγκεκριμένη μελέτη, παρουσιάζονται εκτός των άλλων, κώδικες μαθητών που αντανακλούν την κατανόηση τους πάνω στις έννοιες του παράλληλου προγραμματισμού. Τα αποτελέσματα έδειξαν πως στους μαθητές του δημοτικού είναι δυνατό να διδαχθεί ο παράλληλος προγραμματισμός και βασικές έννοιες, όπως οι συνθήκες ανταγωνισμού και τα αμοιβαία κλειδώματα καθώς μετά την διδασκαλία οι παράλληλες διεργασίες αναπτύχθηκαν στην υπολογιστική τους σκέψη.

Το 2014 οι Feldhausen, Bell και Andresen δημοσίευσαν μια έρευνα με τα αποτελέσματα της διδασκαλίας των εννοιών παράλληλου προγραμματισμού σε μαθητές γυμνασίου και λυκείου, χρησιμοποιώντας την πλατφόρμα προγραμματισμού Scratch στα πλαίσια του STEM. Η μελέτη έδειξε ότι οι μαθητές, ακόμα και αρχάριοι στον προγραμματισμό, ήταν

ικανοί να καταλάβουν τι είναι ο παράλληλος προγραμματισμός και να μπορούν να τον εφαρμόσουν σε μελλοντικά προβλήματα που θα τους παρουσιαστούν. Χρησιμοποιώντας το προγραμματιστικό περιβάλλον του Scratch οι εκπαιδευόμενοι ήταν σε θέση να δημιουργήσουν και να τροποποιήσουν παράλληλα προγράμματα αλλά και να κατανοήσουν τη διαφορά στην απόδοση των προγραμμάτων όταν αυτά έχουν παράλληλο κώδικα μέσα τους. Στη συγκεκριμένη διδασκαλία, προσεγγίστηκαν ακόμα και πιο δύσκολα θέματα παραλληλίας όπως ο ανταγωνισμός για πόρους και τα αμοιβαία κλειδώματα. Συμπερασματικά, οι εκπαιδευόμενοι απέκτησαν ενδιαφέρον για τον παράλληλο προγραμματισμό αλλά και ανέπτυξαν την αυτο-αποτελεσματικότητα τους.

Επιπρόσθετα, ο Pellas (2014) χρησιμοποιώντας τον εικονικό κόσμο OpenSim και την πλατφόρμα προγραμματισμού Scratch ερεύνησε εάν οι εργασίες που δημιουργήθηκαν σε αυτές τις πλατφόρμες εκτός των άλλων, συμβάλλουν στην δημιουργία κινήτρων για καλύτερη κατανόηση και εκμάθηση βασικών αλγοριθμικών εντολών παράλληλου και οπτικού προγραμματισμού σε μαθητές λυκείου. Η συμμετοχή των μαθητών σε αυτά τα μαθήματα προγραμματισμού φάνηκε να έχει θετικά αποτελέσματα για τις παράλληλες διεργασίες στην υπολογιστική τους σκέψη. Πιο συγκεκριμένα, οι εκπαιδευμένοι απάντησαν θετικά στο αν αντιμετώπισαν με ευχαρίστηση την εκμάθηση και τη χρησιμοποίηση εντολών παράλληλου προγραμματισμού μέσω της συνεργατικής διαδικασίας που παρέχουν τα προγραμματιστικά αυτά περιβάλλοντα.

Το 2016 δημοσιεύθηκε μια έρευνα από τους Sáez López, Román-González και Vázquez-Cano που αφορά την ανάλυση των αποτελεσμάτων για την εκμάθηση προγραμματιστικών εννοιών, της λογικής και υπολογιστικής σκέψης σε 107 μαθητές πέμπτης και έκτης δημοτικού στην Ισπανία. Η πλατφόρμα που χρησιμοποιήθηκε ήταν το Scratch και το διάστημα συλλογής δεδομένων ήταν δύο ακαδημαϊκά χρόνια. Όσον αφορά τον παράλληλο προγραμματισμό φάνηκε πως είναι δυνατή η εκμάθησή του από τους μαθητές της πρωτοβάθμιας εκπαίδευσης μαζί με άλλες προγραμματιστικές έννοιες όπως η ακολουθία, η επανάληψη, τα γεγονότα και η κοινοχρησία περιεχομένου. Πιο συγκεκριμένα, οι μαθητές δημιούργησαν τμήματα κώδικα που εκτελέστηκαν παράλληλα με επιτυχία. Μέσω των δεδομένων που συλλέχθηκαν φάνηκε ότι κατά την διάρκεια της διαδικασίας, υπήρχε στους εκπαιδευόμενους το συναίσθημα της ευχαρίστησης αλλά και κλίμα συνεργασίας.

Οι Fatourou, Zygoyris και Loykopoulos και Stamoulis (2018) δημοσίευσαν μια μελέτη που αφορά την αξιολόγηση της πρώιμης εισαγωγής εννοιών παράλληλου προγραμματισμού σε μαθητές δημοτικού, δώδεκα και έντεκα ετών. Τα μαθήματα έγιναν σε 74 μαθητές, διήρκησαν 12 εβδομάδες και χρησιμοποιήθηκε η πλατφόρμα του Scratch. Τα αποτελέσματα έδειξαν πως οι μαθητές μπορούν να σκεφτούν παράλληλα, με τους περισσότερους στόχους να καλύπτονται με επιτυχία και τους δωδεκάχρονοι μαθητές, με λίγη εμπειρία στο Scratch, να έχουν καλύτερα αποτελέσματα από τους εντεκάχρονους. Κάποια προβλήματα που αντιμετώπισαν οι μαθητές αφορούσαν τον συγχρονισμό και τις συνθήκες ανταγωνισμού κάτι που είναι σύνηθες όταν οι εκπαιδευόμενοι αναπτύσσουν κώδικα που μπορεί να εκτελεστεί παράλληλα στο Scratch. Μαζί με την προσέγγιση στον παράλληλο προγραμματισμό, προσεγγίστηκαν και κλασικές έννοιες δομημένου προγραμματισμού. Το πολύ σημαντικό στοιχείο της έρευνας είναι πως οι μαθητές μπορούν να σκεφτούν παράλληλα, κάτι το οποίο ακόμη αποδείχθηκε από την σωστή συμπεριφορά των προγραμμάτων στις συνθήκες ανταγωνισμού, χωρίς να υπάρξουν ανεξήγητες συμπεριφορές. Επίσης, οι ερευνητές προτείνουν πως θα ήταν χρήσιμο να γίνεται εισαγωγή των εννοιών παράλληλου προγραμματισμού στο δημοτικό σχολείο.

Οι Weintrop, Hansen, Harlow και Franklin (2018) σε έρευνά τους εξετάζουν εκτός των άλλων, τη χρήση στρατηγικών βασισμένων σε γεγονότα για την παραγωγή παράλληλων τμημάτων κώδικα, σε προγράμματα μαθητών ηλικίας εννέα έως δώδεκα ετών , στο περιβάλλον οπτικού προγραμματισμού Scratch. Στους κώδικες 106 από τους 135 μαθητές χρησιμοποιήθηκε η παραλληλία και αυτό φαίνεται από την ύπαρξη πολλαπλών τμημάτων κώδικα συνδεδεμένων σε ένα γεγονός. Χαρακτηριστικό παράδειγμα αποτελεί η εκτέλεση πολλών τμημάτων κώδικα αφού πυροδοτηθούν από ένα συγκεκριμένο γεγονός. Η σωστή εκτέλεση τμημάτων κώδικα παράλληλα επιτεύχθηκε ευκολότερα με τα απλά μπλοκ αναμονής παρά μέσω της ανταλλαγής μηνυμάτων, κάτι το οποίο όμως εγκυμονεί κινδύνους για την αποτελεσματικότητα του κώδικα. Πολλές φορές, οι μαθητές χρησιμοποίησαν παράλληλες τεχνικές προγραμματισμού χωρίς να το κάνουν επίτηδες.

Τέλος, οι Libert και Vanhoof (2019) δημοσίευσαν μια έρευνά τους που αφορά την ανάλυση των πρώιμων αντιλήψεων μαθητών δώδεκα με δεκαπέντε ετών, για το τι πιστεύουν ότι είναι “παράλληλος προγραμματισμός” χωρίς να έχουν διδαχθεί για αυτόν. Αυτή η έρευνα πραγματοποιήθηκε καθώς για τη διδασκαλία του παράλληλου προγραμματισμού μέσω ενός

κονστρουκτιβιστικού πλαισίου κρίνεται σημαντική η επίγνωση των πρώιμων αντιλήψεων των εκπαιδευόμενων. Από τους εβδομήντα επτά μαθητές που συμμετείχαν στην έρευνα μόνο οι δύο φάνηκε να έχουν μια σωστή διαίσθηση για το τι είναι ο παράλληλος προγραμματισμός με πολλούς από αυτούς να απαντούν ότι είναι όταν δύο προγράμματα έχουν τον ίδιο στόχο. Επίσης, από τους μαθητές ζητήθηκε να δώσουν λύσεις σε ένα πρόβλημα συγχρονισμού. Από τις λύσεις που δόθηκαν εντοπίστηκαν οι πρώιμες αντιλήψεις των εκπαιδευόμενων. Κατά αυτές, ο ηλεκτρονικός υπολογιστής είναι ένα μαύρο κουτί που επιλύει κάθε παράλληλη διεργασία, δεν υπάρχει χρονική καθυστέρηση μεταξύ του πατήματος ενός κουμπιού και της επεξεργασίας των πληροφοριών, αν ο ηλεκτρονικός υπολογιστής είναι γρήγορος επιλύει τα ζητήματα συγχρονισμού, λιγότερη παραλληλία είναι ευκολότερα διαχειρίσιμη και τέλος, με την μεταξύ επικοινωνία των μερών του συστήματος μπορεί να επιτευχθεί σωστά η παραλληλία.

Από την παραπάνω επισκόπηση φαίνεται πως οι μαθητές είναι δυνατό να σκεφτούν παράλληλα αφού διδαχθούν έννοιες παράλληλου προγραμματισμού σε νεαρές ηλικίες δημοτικού (Gregg et al., 2012· Sáez-López et al., 2016· Fatourou et al., 2018· Weintrop et al., 2018) αλλά και γυμνασίου-λυκείου (Feldhausen et al., 2014· Pellas, 2014) με επιτυχία. Μάλιστα, έχουν ιδέες που παραπέμπουν στην παράλληλη επεξεργασία και υπάρχει πιθανότητα να την προτιμούν σε σχέση με τη δομή επανάληψης (Tarkan et al., 2010). Επιπρόσθετα, οι μαθητές παράγουν παράλληλα τμήματα κώδικα χωρίς να το καταλαβαίνουν (Weintrop et al., 2018). Ακόμη και μαθητές αρχάριοι στον προγραμματισμό δείχνουν ικανοί να καταλάβουν τι είναι ο παράλληλος προγραμματισμός και να τον εφαρμόσουν σε επίπεδο αρχάριου (Feldhausen et al., 2014· Fatourou, 2018). Παρόλα αυτά, από την ανασκόπηση προκύπτει πως οι ανήλικοι μαθητές δεν έχουν αντίληψη για το τι μπορεί να είναι ο παράλληλος προγραμματισμός εάν δεν έχουν διδαχθεί για αυτόν (Libert & Vanhoof, 2019). Συνολικά, από τις παραπάνω έρευνες και μελέτες, φαίνεται να υπάρχουν παράλληλες διεργασίες στην υπολογιστική σκέψη των μαθητών του δημοτικού, οι οποίες ενισχύονται με την διδασκαλία των εννοιών και της εφαρμογής του παράλληλου προγραμματισμού.

Τα ελάχιστα, αλλά ενδιαφέροντα ευρήματα της επισκόπησης αναδεικνύουν την αξία διερεύνησης εμφάνισης παράλληλου προγραμματισμού σε κώδικες μαθητών που βρίσκονται στα πρώτα βήματα διδασκαλίας θεμάτων προγραμματισμού.

1.7. Κατηγοριοποίηση τοπολογιών παράλληλου προγραμματισμού στο Scratch με χρήση της ταξινομίας SOLO

1.7.1. Τοπολογίες παράλληλου προγραμματισμού στο Scratch

Οι οπτικές γλώσσες προγραμματισμού, που η εκτέλεση κάποιων διεργασιών είναι βασισμένη σε γεγονότα, δίνουν την δυνατότητα για την ύπαρξη παράλληλων τμημάτων κώδικα (Weintrop et al., 2018). Στα περιβάλλοντα αυτά, πολλαπλά αντικείμενα μπορούν να εκτελούνται παράλληλα σαν ανταπόκριση στο ίδιο γεγονός. Η δυνατότητα αυτή για παραλληλία υπάρχει και σε γλώσσες προγραμματισμού διαφορετικού υποδείγματος, αλλά αυτή η μορφή παραλληλίας δεν απευθύνεται σε αρχάριους προγραμματιστές.

Στο προγραμματιστικό περιβάλλον οπτικού προγραμματισμού Scratch υπάρχει η δυνατότητα για παράλληλη εκτέλεση παραπάνω από δύο τμημάτων κώδικα (σενάρια) (Λαδιάς, Καρβουνίδης & Λαδιάς, 2020). Αυτή η εκτέλεση συμβαίνει ψευδοπαράλληλα καθώς γίνεται από έναν επεξεργαστή μέσω μηχανισμών της γλώσσας, ο οποίος εναλλάσσει ταχύτατα την λειτουργία του από την εκτέλεση μιας διεργασίας σε μια άλλη και ξανά το ίδιο. Οι διεργασίες στο Scratch μπορούν να ενεργοποιηθούν είτε από εξωτερικούς παράγοντες όπως είναι η ανθρώπινη αλληλεπίδραση είτε από εντολές του ίδιου του κώδικα όπως είναι οι εντολές αποστολής μηνυμάτων και οι εντολές δημιουργίας κλώνων.

Μια διεργασία μπορεί να βρεθεί σε τρεις διαφορετικές καταστάσεις: την ενεργή κατάσταση, την κατάσταση αδράνειας και την κατάσταση λήξης. Κατά την ενεργή κατάσταση μια διεργασία εκτελείται ή είναι σε επαγρύπνηση μέχρι να πυροδοτήθει από κάτι, κατά την κατάσταση αδράνειας μια διεργασία αναμένει την ολοκλήρωση από κάτι ώστε να συνεχίσει την εκτέλεσή της και κατά την κατάσταση λήξης, η διεργασία έχει ολοκληρωθεί.

Οι παράλληλες εκτελέσεις των σεναρίων στο περιβάλλον οπτικού προγραμματισμού του Scratch, μπορούν είτε να ξεκινήσουν ταυτόχρονα από την αρχή του προγράμματος, είτε να δημιουργηθούν στη συνέχεια, κατά την εκτέλεση ενός σειριακού προγράμματος (Λαδιάς, Καρβουνίδης & Λαδιάς, 2020). Τα παράλληλα αυτά τμήματα κώδικα, είτε εκτελούνται εξ ολοκλήρου ταυτόχρονα, είτε μπορούν να ενεργοποιηθούν από ξεχωριστά συμβάντα, και να βρεθεί ένα τμήμα του κώδικα να εκτελείται ταυτόχρονα με ένα άλλο τμήμα του κώδικα. Μια διεργασία που πυροδοτεί ένα άλλο τμήμα κώδικα, μπορεί να τερματίσει μετά την πυροδότηση ή να συνεχίσει να εκτελείται παράλληλα με τη διεργασία απόγονο που

δημιούργησε. Επίσης, παραλληλία υπάρχει όταν η ίδια διεργασία εκτελείται πολλές φορές ταυτόχρονα. Τα παράλληλα σενάρια μπορούν να ολοκληρωθούν, είτε όταν ένα από αυτά ολοκληρωθεί και διακόψει τα υπόλοιπα, είτε όταν τερματίσουν όλα αφού τελείωσαν.

Στο περιβάλλον του Scratch όταν υπάρχει παραλληλία δεν παύουν να ελλοχεύουν τα προβλήματα και οι κίνδυνοι που περιγράφουν τον παράλληλο προγραμματισμό (Λαδιάς, Καρβουνίδης & Λαδιάς, 2020). Το βασικό πρόβλημα του παράλληλου προγραμματισμού είναι μη αιτιοκρατική συμπεριφορά ενός προγράμματος με παράλληλα τμήματα κώδικα. Αυτή η συμπεριφορά αναφέρεται στο ότι ένα παράλληλο πρόγραμμα παίρνοντας την ίδια είσοδο δίνει διαφορετική έξοδο καθώς ακολουθεί διαφορετικά μονοπάτια εκτέλεσης. Αυτό συμβαίνει γιατί πολλαπλές διεργασίες ανταγωνίζονται για την πρόσβασή τους σε τμήματα κώδικα τα οποία ονομάζονται κρίσιμα, με αποτέλεσμα να συγκρούονται και κάθε φορά να είναι απρόβλεπτο το πιο τμήμα θα πάρει την πρόσβαση. Για την αντιμετώπιση αυτού του προβλήματος προκύπτει η ανάγκη συγχρονισμού. Αυτό μπορεί να γίνει σε ένα πρώτο στάδιο με την εντολή «περίμενε ώσπου…». Λόγω της νεαρής ηλικίας των μαθητών που απευθύνεται το Scratch, πιο εξεζητημένα θέματα παράλληλου προγραμματισμού όπως οι σηματωροί, δεν προσεγγίζονται με ευκολία.

Αυτά τα στοιχεία παράλληλου προγραμματισμού των οπτικών γλωσσών προγραμματισμού και συγκεκριμένα του Scratch, δίνουν τη δυνατότητα για αξιολόγηση κώδικα μαθητών ως προς την παραλληλία.

1.7.2. Ταξινομία SOLO

Κατά τους Biggs & Collins (1982) η γνώση μπορεί να αξιολογηθεί με βάση τη δομή του παρατηρούμενου μαθησιακού αποτελέσματος και κατ’ επέκταση πρότειναν την ταξινομία SOLO (Structure of the Observed Learning Outcome). Η ταξινομία αυτή είναι ένα μοντέλο που περιγράφει πέντε επίπεδα της πολυπλοκότητας της κατανόησης ενός γνωστικού αντικειμένου. Το πρώτο επίπεδο είναι το προδομικό (**prestructura**l) στο οποίο γίνεται αναφορά ή χρήση μη συνδεδεμένων και ανοργάνωτων πληροφοριών που δεν έχουν νόημα. Σε αυτό το επίπεδο ουσιαστικά ο εκπαιδευόμενος δεν έχει κατανοήσει το νόημα και το πώς να χρησιμοποιεί την πληροφορία που διδάσκεται. Το δεύτερο επίπεδο είναι το μονοδομικό (**unistructural**) και σε αυτό παρατηρείται μια περιορισμένη οπτική - κυρίως χρησιμοποιείται ή τονίζεται ένα στοιχείο ή μια πτυχή - ενώ παραλείπονται οι υπόλοιπες συνιστώσες και δεν

πραγματοποιούνται σημαντικές συνδέσεις μεταξύ των μερών. Το τρίτο επίπεδο είναι το πολυδομικό (**multistructural**), στο οποίο υπάρχει μια προοπτική πολλαπλών σημείων - χρησιμοποιούνται ή αναγνωρίζονται διάφορα σχετικά στοιχεία ή πτυχές - αλλά δεν υπάρχουν σημαντικές συνδέσεις και δεν έχει διαμορφωθεί ακόμη μια ολοκληρωμένη εικόνα. Το τέταρτο επίπεδο είναι το σχεσιακό (**relational**), στο οποίο υπάρχει μια ολιστική προοπτική όπου οι μετα-συνδέσεις μεταξύ των μερών γίνονται αντιληπτές και η σημασία των τμημάτων σε σχέση με το σύνολο αποδεικνύεται και εκτιμάται. Τέλος, το πέμπτο επίπεδο είναι αυτό της εκτεταμένης γενίκευσης (**extended abstract**), κατά την οποία επιπλέον των χαρακτηριστικών της προηγούμενης σχεσιακής κατηγορίας, το περιεχόμενο αντιμετωπίζεται ως ένα στιγμιότυπο μιας γενικότερης περίπτωσης.

1.7.3. Τοπολογίες παράλληλου προγραμματισμού και επίπεδα της SOLO

Το 2021 οι Λαδιάς και Καρβουνίδης κατηγοριοποίησαν με τη χρήση της ταξινομίας SOLO τις τοπολογίες παράλληλου προγραμματισμού στο περιβάλλον του Scratch με σκοπό τη δημιουργία μιας βάσης για την αξιολόγηση του παράλληλου κώδικα. Παρακάτω, γίνεται παρουσίαση της κατηγοριοποίησης αυτής και χρησιμοποιούνται τα βοηθητικά σχήματα που χρησιμοποιούν οι Λαδιάς και Καρβουνίδης (2021).

Στο προδομικό επίπεδο εντάσσονται οι κώδικες των εκπαιδευόμενων που είναι σειριακοί και δεν έχουν παράλληλες διεργασίες όπως φαίνεται στο σχήμα 1.7.1. Εκτός αυτών, σε αυτό το επίπεδο κατηγοριοποιούνται οι κώδικες στους οποίους υπάρχουν παράλληλες διεργασίες όμως ο τρόπος που χρησιμοποιούνται δείχνει πως οι εκπαιδευόμενοι έχουν πολύ περιορισμένη επίγνωση της διαχείρισής τους και κατ' επέκταση δεν λαμβάνουν υπόψη την μη αιτιοκρατική συμπεριφορά τους. Στο σχήμα 1.7.2, οι διεργασίες δεν λαμβάνουν υπόψη πως πρέπει να προηγηθεί η αρχικοποίηση της μεταβλητής **μ**, πριν την αλλαγή της τιμής της οπότε δεν λαμβάνεται υπόψη η μη αιτιοκρατική συμπεριφορά τους.

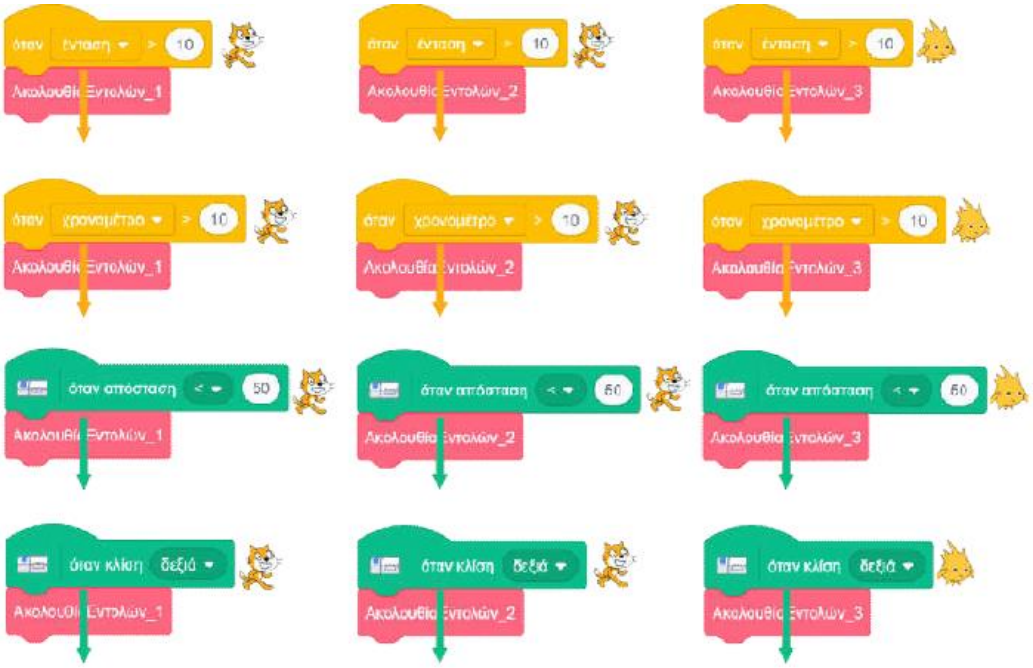
Σχήμα 1.7. 1 Σειριακός κώδικας χωρίς παράλληλες διεργασίες

Σχήμα 1.7. 2 Μη αιτιοκρατική συμπεριφορά παράλληλων διεργασιών

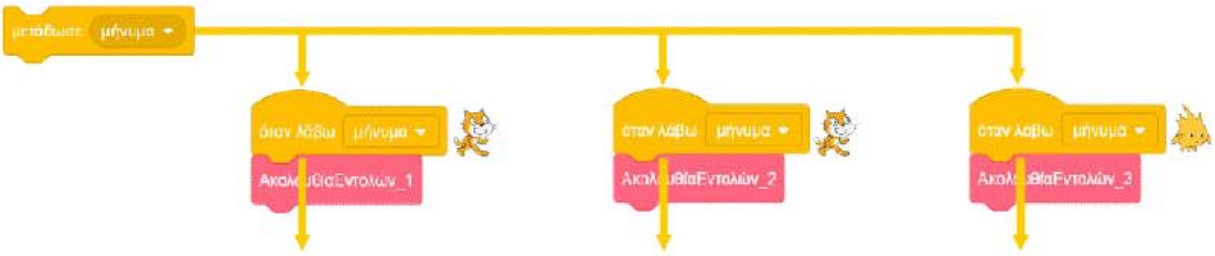
Στο μονοδομικό επίπεδο εντάσσονται οι κώδικες, με διεργασίες που λειτουργούν ανεξάρτητα και ξεκινούν ταυτόχρονα, ενεργοποιούμενες από την ίδια αιτία που μπορούν να κατανέμονται σε έναν ή σε διαφορετικούς ρόλους αντικειμένων όπως δείχνει το Σχήμα 1.7.3. Σε αυτό το επίπεδο οι διεργασίες δεν έχουν την ανάγκη να επικοινωνούν ή να συγχρονίζονται μεταξύ τους. Οι διεργασίες αυτές, ως αιτία πυροδότησης, μπορεί να έχουν εξωτερικά συμβάντα όπως αλληλεπίδραση με το χρήστη μέσω μικροφώνου, χρονομέτρου, αισθητήρα απόστασης, αισθητήρα κλίσης κ.α. (Σχήμα 1.7.4) είτε εσωτερικά συμβάντα όπως είναι η μετάδοση κάποιου μηνύματος (Σχήμα 1.7.5). Στο Scratch, αιτία δημιουργίας παράλληλων

διεργασιών με ταυτόχρονη εκκίνηση, μπορεί να αποτελέσει και η δημιουργία κλώνων (Σχήμα 1.7.6). Τέλος, κατά την ένταξη του κώδικα στο μονοδομικό ή προδομικό επίπεδο, παίζει ρόλο κατά πόσο ο εκπαιδευόμενος εντοπίζει μια λύση στην οποία χρησιμοποιεί παραλληλία και δεν χρησιμοποιεί μια σειριακή λύση.

Σχήμα 1.7. 3 Ανεξάρτητες παράλληλες διεργασίες με ταυτόχρονη εκκίνηση από την ίδια αιτία



Σχήμα 1.7. 4 Εξωτερικά συμβάντα ως αιτία πυροδότησης παράλληλων διεργασιών στο Scratch

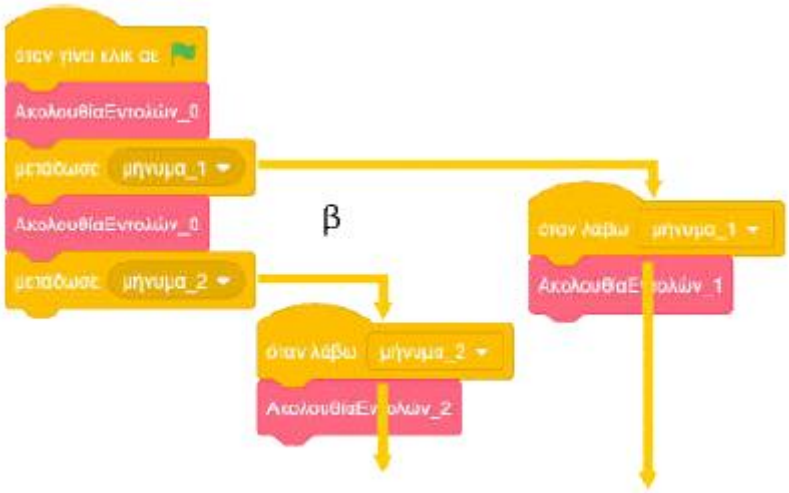


Σχήμα 1.7. 5 Εσωτερικά συμβάντα ως αιτία πυροδότησης παράλληλων διεργασιών στο Scratch

Σχήμα 1.7. 6 Δημιουργία κλώνων ως αιτία πυροδότησης παράλληλων διεργασιών στο Scratch

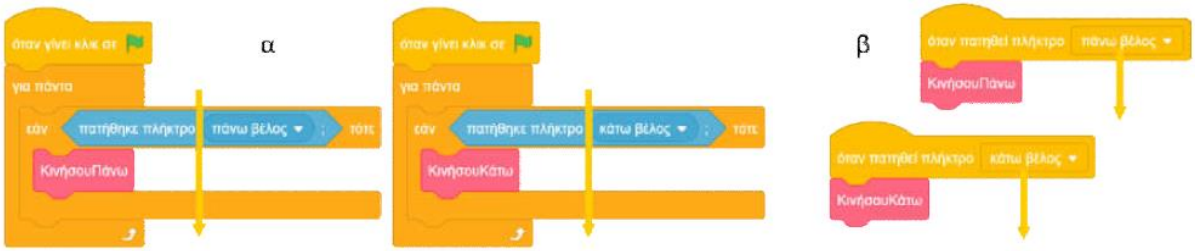
Στο πολυδομικό επίπεδο τα προγράμματα χαρακτηρίζονται από παράλληλες διεργασίες, οι οποίες λειτουργούν ανεξάρτητα και δεν ξεκινούν ταυτόχρονα καθώς οι πηγές ενεργοποίησης είναι διαφορετικές, είτε εξωτερικά συμβάντα (αλληλεπίδραση με το χρήστη, συσκευές) είτε εσωτερικά συμβάντα (Σχήμα 1.7.7), όπως η μετάδοση μηνυμάτων ή δημιουργία κλώνων. Χαρακτηριστικό παράδειγμα αυτού του επιπέδου είναι η διαδοχική ενεργοποίηση διεργασιών (Σχήμα 1.7.8). Οι διεργασίες αυτές δεν έχουν την ανάγκη να επικοινωνούν μεταξύ τους και να συγχρονίζονται και μπορούν να κατανέμονται στον ίδιο ή σε διαφορετικούς ρόλους αντικειμένων. Για την ένταξη του προγράμματος στο μονοδομικό ή πολυδομικό επίπεδο συνεκτιμάται αν ο εκπαιδευόμενος μπορεί να εντοπίσει μια λύση κατά την ανάπτυξη του κώδικά του που οι παράλληλες διεργασίες ξεκινούν ταυτόχρονα (μονοδομικό επίπεδο) από μια λύση που οι παράλληλες διεργασίες δεν αρχίζουν ταυτόχρονα και η λύση εντάσσεται στο πολυδομικό επίπεδο. Επιπλέον, λύσεις με κώδικες που βρίσκονται σε κατάσταση εκτέλεσης σε μια διαρκή επανάληψη εντάσσονται στο μονοδομικό επίπεδο, ενώ λύσεις με κώδικες που δεν εκτελούνται συνεχώς αλλά βρίσκονται σε μια κατάσταση επαγρύπνησης εντάσσονται στο πολυδομικό επίπεδο (Σχήμα 1.7.9).

Επιπρόσθετα, στο πολυδομικό επίπεδο ανήκουν οι παράλληλες διεργασίες που δεν ολοκληρώνονται εκτός και αν ολοκληρωθεί και η τελευταία από αυτές. Τέλος, όταν μια πατρική διεργασία συνεχίζει τη λειτουργία της παράλληλα με τις διεργασίες απογόνους, ο κώδικας εντάσσεται στο πολυδομικό επίπεδο (Σχήμα 1.7.10).

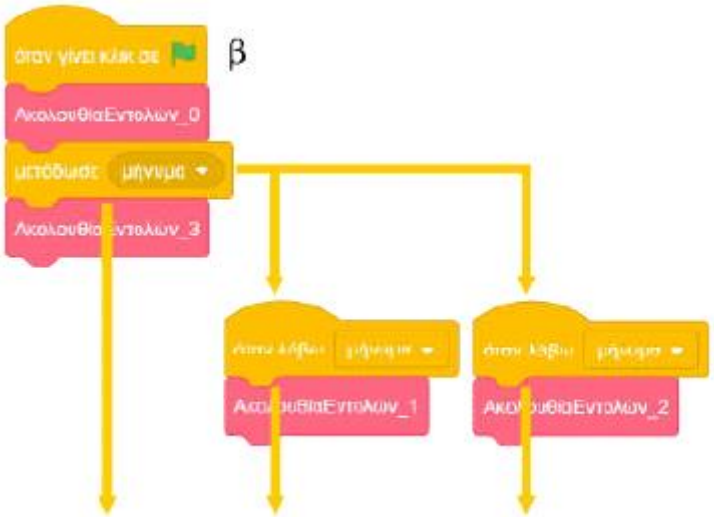


Σχήμα 1.7. 7 Ανεξάρτητες παράλληλες διεργασίες με μη ταυτόχρονη πυροδότηση από διαφορετική αιτία

Σχήμα 1.7. 8 Διαδοχική ενεργοποίηση διεργασιών με μη ταυτόχρονη πυροδότηση από διαφορετική αιτία



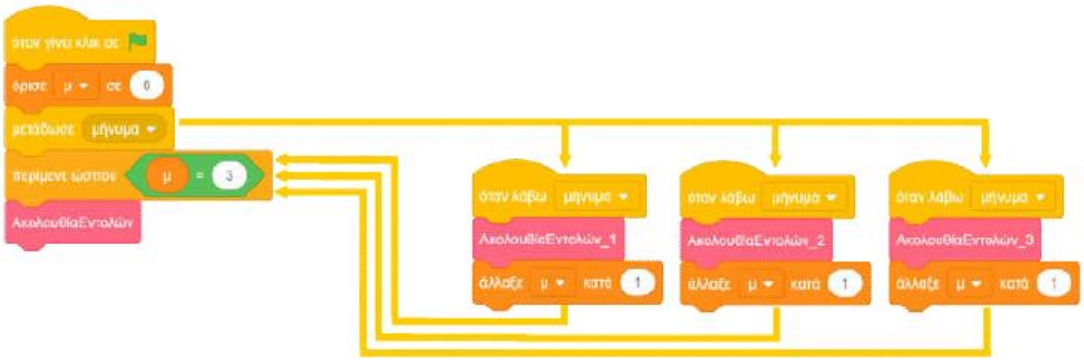
Σχήμα 1.7. 9 (α) Παράλληλες διεργασίες σε μόνιμη κατάσταση εκτέλεσης - (β) Παράλληλες διεργασίες σε κατάσταση επαγρύπνησης



Σχήμα 1.7. 10 Πατρική διεργασία συνεχίζει να εκτελείται παράλληλα με τις διεργασίες απογόνους μετά την πυροδότηση τους

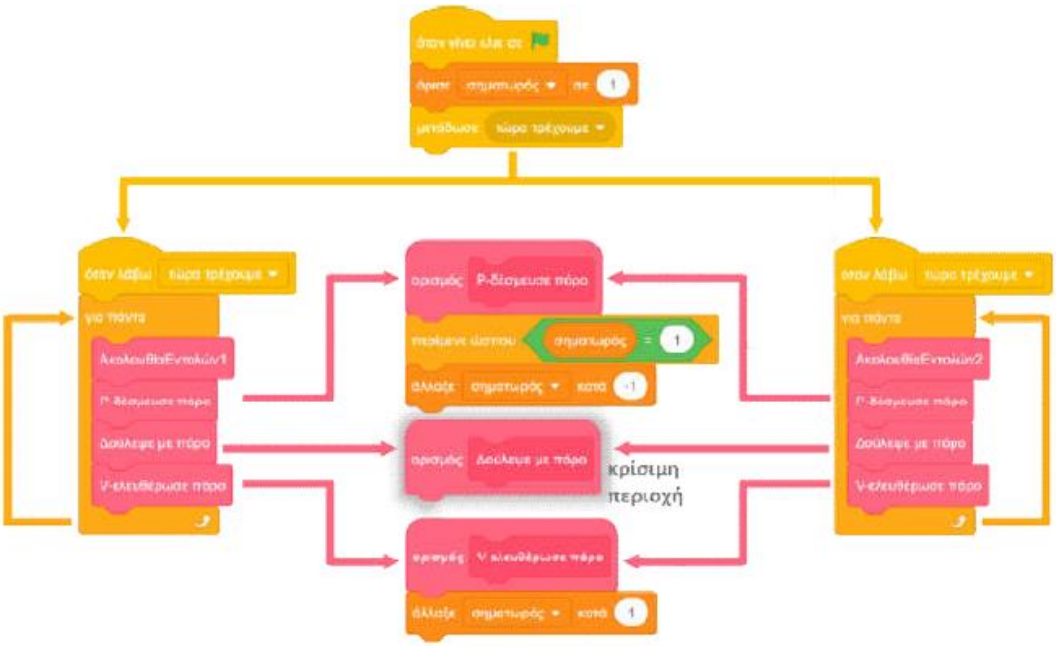
Στο σχεσιακό επίπεδο της ταξινομίας SOLO εντάσσονται οι κώδικες με παράλληλες διεργασίες που εξαρτώνται η μία από την άλλη και έχουν την ανάγκη να επικοινωνούν. Στο Scratch η εντολή “μετάδωσε… και περίμενε” επιτρέπει την εκτέλεση των εντολών που βρίσκονται παρακάτω στο πατρικό σενάριο, μόνο εάν τερματίσουν όλες οι παράλληλες διεργασίες που πυροδοτούνται από αυτό το μήνυμα (Σχήμα 1.7.11). Αυτός ο τρόπος συγχρονισμού ονομάζεται σύγχρονη επικοινωνία. Ένας άλλος τρόπος επικοινωνίας, είναι μέσω διαμοιρασμού της μνήμης που χρησιμοποιεί καθολικές μεταβλητές στις οποίες έχουν πρόσβαση όλα τα αντικείμενα του κώδικα (Σχήμα 1.7.12). Πιο συγκεκριμένα, κάθε διεργασία αποθηκεύει προσωρινά σε μια καθολική μεταβλητή τα δεδομένα ώστε κάθε άλλη διεργασία να έχει τη δυνατότητα προσπέλασης σε αυτά. Αυτή η μορφή επικοινωνίας ονομάζεται ασύγχρονη.

Σχήμα 1.7. 11 Σύγχρονη επικοινωνία διεργασιών



Σχήμα 1.7. 12 Ασύγχρονη επικοινωνία διεργασιών

Στο επίπεδο εκτεταμένης γενίκευσης εντάσσονται οι κώδικες που κατά τη δημιουργία τους λήφθηκε υπόψη η μη αιτιοκρατική συμπεριφορά του κώδικα και ο προγραμματιστής επίλυσε προβλήματα, όπως η χωρίς αμοιβαίο αποκλεισμό πρόσβαση σε κρίσιμες περιοχές κώδικα διαχείρισης κοινόχρηστου πόρου. Χαρακτηριστικό παράδειγμα είναι όταν ένα σενάριο τρέχει παράλληλα με τον εαυτό του, και αυτά τα παράλληλα τμήματα κώδικα εκτελούν τον ίδιο κώδικα αλλά με διαφορετικά δεδομένα κάθε φορά, με αποτέλεσμα το πρόγραμμα να βρεθεί σε απροσδιόριστες και ανεξέλεγκτες καταστάσεις. Ένας τρόπος επίλυσης ζητημάτων διεκδίκησης της πρόσβασης σε μια κρίσιμη περιοχή κώδικα είναι οι σηματωροί. Αυτοί είναι απλές διαμοιραζόμενες μεταβλητές, που σκοπό έχουν την προστασία ενός κοινόχρηστου πόρου καθώς διαχειρίζονται την πρόσβαση ή την απεμπλοκή μιας παράλληλης διεργασίας σε μια κρίσιμη περιοχή κώδικα. Το Scratch παρέχει τη δυνατότητα χρησιμοποίησης των σηματωρών για τον έλεγχο πρόσβασης σε μια κρίσιμη περιοχή από πολλαπλές παράλληλες διεργασίες (Σχήμα 1.7.13).



Σχήμα 1.7. 13 Έλεγχος πρόσβασης σε κρίσιμη περιοχή κώδικα από πολλαπλές διεργασίες με τη χρήση σηματωρών

2.ΕΡΕΥΝΗΤΙΚΟ ΜΕΡΟΣ

2.1 Μεθοδολογία

2.1.1. Σκοπός έρευνας

Σκοπός της παρούσας μελέτης είναι να διερευνηθεί κατά πόσο υπάρχουν παράλληλες διεργασίες στην υπολογιστική σκέψη μαθητών δέκα έως δεκατριών ετών, ελέγχοντας με μετρήσιμα κριτήρια τον τρόπο με τον οποίο χρησιμοποιούν τον παράλληλο προγραμματισμό σε προγράμματά τους, τα οποία ανέπτυξαν για να ελέγξουν ρομποτικές κατασκευές.

Από την βιβλιογραφική ανασκόπηση προκύπτει πως φαίνεται να μπορούν να υπάρξουν αυτές οι διεργασίες στην υπολογιστική σκέψη των μαθητών του δημοτικού. Μάλιστα, όταν υπάρξει διδασκαλία εννοιών παράλληλου προγραμματισμού οι μαθητές ανταποκρίνονται καλύτερα. Οι παρόμοιες έρευνες που έχουν γίνει είναι πολύ μικρές σε αριθμό και κάποιες φορές εξετάζουν ταυτόχρονα και την επίδοση των μαθητών σε άλλες πτυχές του προγραμματισμού. Καταλήγοντας, με κριτήριο τις παρελθοντικές έρευνες η ερευνητική υπόθεση είναι πως υπάρχουν κάποιας μορφής παράλληλες διεργασίες στην υπολογιστική σκέψη των μαθητών.

2.1.2. Ανάλυση δεδομένων

Για το ερευνητικό σκέλος της εργασίας ακολουθείται μια μεθοδολογία βασισμένη στην ανάλυση περιεχομένου. Σύμφωνα με τον Krippendorff (2004), η ανάλυση περιεχομένου είναι «μια τεχνική έρευνας με σκοπό την εξαγωγή έγκυρων συμπερασμάτων από ένα κείμενο ή από οποιονδήποτε τρόπο επικοινωνίας ο οποίος περιέχει νοήματα, στο πλαίσιο χρήσης του». Η τεχνική ανάλυσης περιεχομένου αποτελείται από μια πληθώρα βημάτων με το πρώτο από αυτά να είναι ο προσδιορισμός του σκοπού της έρευνας και η διατύπωση ερευνητικών ερωτημάτων ή υποθέσεων. Στη συνέχεια, ακολουθεί ο καθορισμός και η επιλογή του πληθυσμού και του δείγματος. Το τρίτο βήμα αφορά τον προσδιορισμό της μονάδας ανάλυσης και των μεταβλητών. Κατά το τέταρτο βήμα, λαμβάνει χώρα η κατηγοριοποίηση των δεδομένων. Τέλος, ακολουθεί η ανάλυση των δεδομένων και η εξαγωγή συμπερασμάτων. Ακολούθως, αναλύονται τα προαναφερθέντα βήματα για την παρούσα εργασία.

2.1.3. Δείγμα

Στην έρευνα χρησιμοποιήθηκαν ως δείγμα είκοσι πέντε προγράμματα για τον έλεγχο αυτοματισμών και των προσομοιώσεών τους από το αποθετήριο έργων της ανοιχτής Κατηγορίας για το Δημοτικό, του Πανελλήνιου Διαγωνισμού Εκπαιδευτικής Ρομποτικής του WRO - Hellas του 2020. Οι μαθητές που ανέπτυξαν τους κώδικες αυτούς είναι ηλικίας δέκα έως δεκατριών ετών ενώ κάθε κώδικας γράφτηκε από ομάδες τριών έως έξι ατόμων. Το γνωστικό επίπεδο των μαθητών και η επίδοσή τους στα αντικείμενα που διδάσκονται είναι άγνωστο. Επίσης, για την αναπαράσταση των προγραμμάτων χρησιμοποιήθηκαν, τα αντίστοιχα σε αυτά, κωδικΌραματα. Τέλος, τα πρόγραμμα που επιλέχθηκαν είναι αυτά που εκτελούνται με επιτυχία σε συνδυασμό με ένα καλά γραμμένο κωδικΌραμα ώστε να υπάρχει η δυνατότητα για ανάλυση.

2.1.3.1 ΚωδικΌραμα

Το κωδικΌραμα είναι ένας οπτικοποιημένος δισδιάστατος πίνακας που απεικονίζει το σύνολο ενός κώδικα που έχει γραφτεί στην πλατφόρμα οπτικού προγραμματισμού Scratch (Λαδιάς & Λαδιάς, 2016). Προέκυψε από την ανάγκη για την διαχείριση πολύπλοκων κωδίκων στον οπτικό προγραμματισμό, ανεπτυγμένους από μαθητές μικρών ηλικιών καθώς τα τελευταία χρόνια ο προγραμματισμός έχει γίνει κομμάτι του ελληνικού προγράμματος σπουδών. Οι πολύ μεγάλοι κώδικες δημιουργούν προβλήματα στους μαθητές, τα οποία αφορούν κυρίως το πότε ένα αντικείμενο και σε ποια κατάσταση χρησιμοποιεί ποιο τμήμα κώδικα. Μέσω των κωδικΟραμάτων γίνεται προσπάθεια να ξεπεράσει η αδυναμία των λογισμικών οπτικού προγραμματισμού να οργανώσουν τον τμηματοποιημένο κώδικα των διαφόρων αντικειμένων.

Στην οριζόντια γραμμή αυτού του δισδιάστατου πίνακα βρίσκονται όλα τα αντικείμενα που υπάρχουν στον κώδικα ενώ στην κατακόρυφη βρίσκονται οι καταστάσεις που μπορεί να υπάρξουν τα αντικείμενα ενώ το πρόγραμμα εκτελείται. Έτσι δημιουργείται ο πίνακας με διαστάσεις τον αριθμό των αντικειμένων επί τον αριθμό των καταστάσεων. Κάθε κελί περιέχει το τμήμα κώδικα που εκτελεί ένα αντικείμενο στην αντίστοιχη κατάσταση. Πολλά από τα κελιά που δημιουργούνται υπάρχει πιθανότητα να παραμένουν κενά. Στα κωδικΌραματα επίσης, υπάρχουν διασυνδέσεις που απεικονίζονται με βέλη και δείχνουν την κίνηση της πληροφορίας κατά την επικοινωνία των διαφόρων τμημάτων κώδικα. Τα χρώματα που χρησιμοποιούνται για τις διασυνδέσεις είναι το μωβ (Scratch-2) ή το κόκκινο

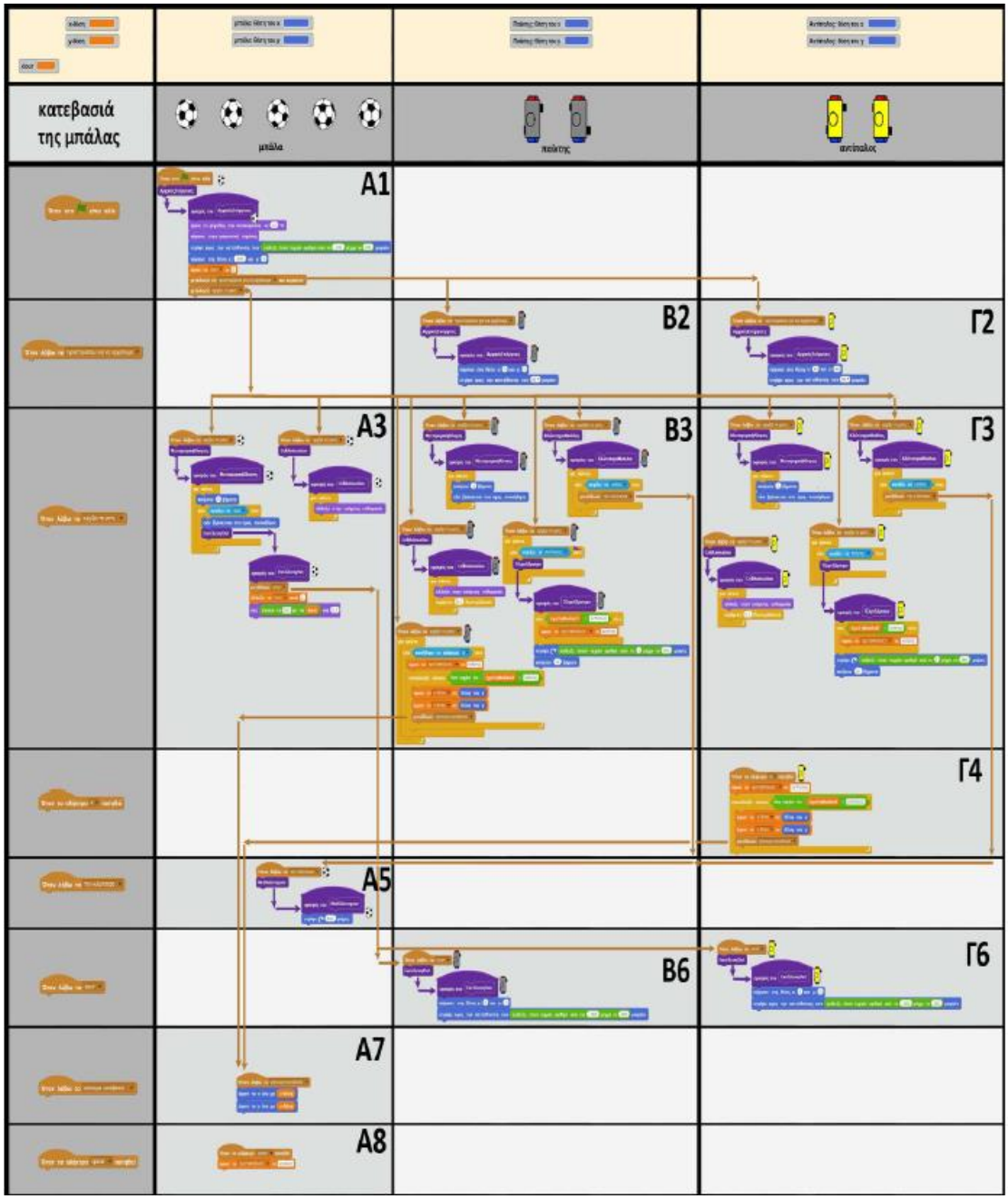
(Scratch-3) για τη διαχείριση διαδικασιών, το πορτοκαλί για τη διαχείριση των μεταβλητών, το καφέ για τη διαχείριση των μηνυμάτων και το μουσταρδί για τη διαχείριση των κλώνων. Οι παραπάνω περιγραφές φαίνονται στο Σχήμα 2.1.1 που αποτελεί ένα ολοκληρωμένο κωδικόΟραμα.

Όσον αφορά την ευαναγνωσιμότητα ενός ΚωδικΟράματος, αυτή εξαρτάται από το εάν ο κώδικας ακολουθεί τις συμβάσεις του ευανάγνωστου προγραμματισμού και την εννοιολογική ονοματοθεσία μεταβλητών και διαδικασιών, ώστε να μην χρειάζονται επεξηγηματικά σχόλια που να περιγράφουν τι αναπαριστά η εκάστοτε μεταβλητή και η εκάστοτε διαδικασία. Επίσης, κρίσιμο σημείο για ένα ευανάγνωστο κωδικόΟραμα αποτελεί μια κατάλληλη χωρική τοποθέτηση των αντικειμένων στην κατακόρυφη στήλη. Όταν αυτή επιτυγχάνεται, μειώνεται ο οπτικός θόρυβος καθώς οι γραμμές διασύνδεσης δεν τέμνονται σε μεγάλο βαθμό, αλλά και έρχονται στην επιφάνεια τα στοιχεία συμμετρίας που προκύπτουν. Επιπλέον, η κατάλληλη χωρική τοποθέτηση των καταστάσεων στις γραμμές επιτυγχάνει και αυτή την μείωση του οπτικού θορύβου αλλά και την εν μέρει ανάδειξη της χρονικής εξέλιξης της εκτέλεσης του κώδικα.

Το κωδικόΟραμα φαίνεται να επιτυγχάνει τον κύριο στόχο για τον οποίο δημιουργήθηκε, δηλαδή να παρέχει βοήθεια στην κατανόηση του ποιος κώδικας αντιστοιχεί σε ποιο αντικείμενο και σε ποια κατάσταση. Επιπλέον, επιτρέπεται η ταυτόχρονη εστίαση στο σύνολο του κώδικα αλλά και στα επιμέρους τμήματα του. Ένα άλλο σημαντικό όφελος των κωδικΟραμάτων που προκύπτει, είναι η απλοποίηση της σύγκρισης μεταξύ του μεγέθους και της πυκνότητας δύο προγραμμάτων, μεταξύ σειριακού και παράλληλου προγραμματισμού και μεταξύ τμημάτων κώδικα που τρέχουν παράλληλα εντός ενός συγκεκριμένου αντικειμένου ή και μεταξύ τμημάτων κώδικα που τρέχουν παράλληλα σε πολλαπλά αντικείμενα.

Στα κωδικόΟραματα υπάρχουν και στοιχεία που φαίνεται να χρήζουν μιας περαιτέρω διερεύνησης. Από αυτά είναι, η διάκριση μεταξύ μηνυμάτων από αντικείμενο σε αντικείμενο και από αντικείμενο σε χρήστη, μεταξύ αναπαραγωγής αντικειμένων και δημιουργίας κλώνων, μεταξύ αντικειμενοστραφούς (Object Oriented Programming) και βασισμένου σε αντικείμενο προγραμματισμού (Object Based Programming) καθώς και μεταξύ καθολικών και τοπικών μεταβλητών. Τα παραπάνω αφορούν κυρίως την απεικόνιση και διαχείριση

δεδομένων σε αντίθεση με τα ισχυρά σημεία των κωδικΟραμάτων που είναι κυρίως αλγοριθμικής φύσης.



Σχήμα 2.1. 1 Ένα παράδειγμα ολοκληρωμένου ΚωδικΟράματος

2.1.4. Προσδιορισμός μονάδας Ανάλυσης

Η **μονάδα ανάλυσης** για την παρούσα έρευνα είναι το κάθε κελί του ΚωδικΟράματος. Στην διαδικασία ανάλυσης γίνεται έλεγχος όλων των κελιών ενός ΚωδικΟράματος. Κάθε κελί εμπεριέχει τον κώδικα (ένα η περισσότερα σενάρια) που εκτελείται από ένα αντικείμενο σε

μια συγκεκριμένη κατάσταση. Στο Σχήμα 2.1.2 είναι σημειωμένο με κόκκινο χρώμα το κελί στην πρώτη γραμμή και στην πρώτη στήλη ενός ΚωδικΟράματος που αποτελεί μια μονάδα ανάλυσης. Για κάθε μονάδα γίνεται έλεγχος αν υπάρχουν περισσότερα του ενός τμήματα κώδικα που εκτελούνται ταυτόχρονα και υπό ποιες συνθήκες, ώστε να εντοπιστούν οι τοπολογίες παράλληλου προγραμματισμού εάν αυτές υπάρχουν. Σε μια ολόκληρη γραμμή (Σχήμα 2.1.3) φαίνεται για κάθε κατάσταση αν εκτελούνται παράλληλα κώδικες διαφορετικών αντικειμένων ή ακόμα και παράλληλοι κώδικες του ίδιου αντικειμένου. Τέλος, σε ένα ολόκληρο ΚωδικΌραμα φαίνονται και όλες οι σχέσεις των τμημάτων κώδικα που τρέχουν ταυτόχρονα ή που πυροδοτούν το ένα το άλλο (Σχήμα 2.1.1). Σε αυτές τις σχέσεις εντοπίζεται η πυροδότηση παράλληλων σεναρίων από διαφορετικές αιτίες αλλά και η σύγχρονη και ασύγχρονη επικοινωνία.

Σχήμα 2.1. 2 Ένα παράδειγμα κελιού ΚωδικΟράματος που αποτελεί μία μονάδα ανάλυσης

Σχήμα 2.1. 3 Μια ολόκληρη γραμμή ΚωδικΟράματος που δείχνει ποια παράλληλα σενάρια εκτελούνται

2.1.5. Κατηγοριοποίηση δεδομένων

Η κατηγοριοποίηση των δεδομένων γίνεται μέσω της αξιολόγησης του κώδικα με κριτήριο την παραλληλία. Ειδικότερα, επιτυγχάνεται με την ένταξη κάθε ΚωδικΟράματος σε ένα από τα πέντε επίπεδα της ταξινομίας SOLO με κριτήριο τις τοπολογίες παράλληλου προγραμματισμού που περιέχει, όπως περιεγράφηκαν από τους Λαδιά και Καρβουνίδη το 2021. Στην ενότητα 1.7.3 έγινε εκτενής και αναλυτική παρουσίαση της ένταξης των διαφόρων τμημάτων κώδικα στις κατηγορίες της ταξινομίας της SOLO και με βάση αυτή γίνεται και η ένταξη των κωδικΟραμάτων στην παρούσα εργασία:

- Στο **προδομικό επίπεδο** εντάσσονται τα προγράμματα που τρέχουν τον κώδικα τους σειριακά, χωρίς να εκτελείται μια διεργασία ταυτόχρονα με κάποια άλλη και τα προγράμματα με κώδικες που δεν δίνουν σημασία στην μη-αιτιοκρατική συμπεριφορά των παράλληλων σεναρίων.
- Στο **μονοδομικό επίπεδο** εντάσσονται οι κώδικες που τρέχουν παράλληλα αλλά πυροδοτούνται από την ίδια αιτία και αρχίζουν την εκτέλεση τους ταυτόχρονα.
- Οι κώδικες που ενεργοποιούνται από διαφορετική αιτία και δεν ξεκινούν ταυτόχρονα αλλά είναι ανεξάρτητοι εντάσσονται στο **πολυδομικό επίπεδο**.
- Στο **σχεσιακό επίπεδο** περιέχονται οι κώδικες που τρέχουν παράλληλα και έχουν την ανάγκη για μεταξύ τους επικοινωνία, καθώς εξαρτάται ο ένας από τον άλλον.
- Τέλος, στο **επίπεδο εκτεταμένης γενίκευσης** εντάσσονται οι κώδικες που ο προγραμματιστής έχει επιλύσει προβλήματα μη-αιτιοκρατικής συμπεριφοράς των παράλληλων τμημάτων, όπως η πρόσβαση σε κρίσιμες περιοχές κώδικα και η διαχείριση κοινόχρηστων πόρων.

Η πραγματοποίηση της κατηγοριοποίησης γίνεται με τον έλεγχο κάθε μονάδας, δηλαδή κάθε κελιού στο δισδιάστατο πίνακα ΚωδικΟράματος για το αν υπάρχουν ή όχι οι εξής τοπολογίες παράλληλου προγραμματισμού:

- Σειριακός κώδικας (Προδομικό)
- Μη- αιτιοκρατικός κώδικας (Προδομικό)
- Ταυτόχρονη εκκίνηση ανεξάρτητων, παράλληλων σεναρίων από την ίδια αιτία (Μονοδομικό)
- Παράλληλα σενάρια που βρίσκονται σε μόνιμη κατάσταση εκτέλεσης (Μονοδομικό)

- Μη ταυτόχρονη εκκίνηση ανεξάρτητων, παράλληλων σεναρίων από διαφορετική αιτία (Πολυδομικό)
- Ταυτόχρονη εκτέλεση του πατρικού σεναρίου με τα σενάρια απογόνους που πυροδότησε (Πολυδομικό)
- Ανάγκη επικοινωνίας των παράλληλων σεναρίων που επιτυγχάνεται με σύγχρονη επικοινωνία (Σχεσιακό)
- Ανάγκη επικοινωνίας των παράλληλων σεναρίων που επιτυγχάνεται με ασύγχρονη επικοινωνία (Σχεσιακό)
- Πρόσβαση σε κρίσιμες περιοχές κώδικα από τα παράλληλα σενάρια με τη χρήση σηματωρών (Εκτεταμένης γενίκευσης).

Για τους κώδικες στην πρώτη γραμμή κάθε ΚωδικΟράματος που τρέχουν ταυτόχρονα δεν λαμβάνεται υπόψη η παραλληλία. Αυτό συμβαίνει γιατί εκεί βρίσκεται η εκκίνηση του προγράμματος και θεωρούμε ότι η παραλληλία δεν εντοπίζεται με πρόθεση του προγραμματιστή αλλά λόγω αρχικοποίησης μεταβλητών, εκκίνησης συσκευών και ανάλογες διαδικασίες αρχικοποίησης. Όταν σε ένα κωδικΌραμα υπάρχουν στοιχεία από διαφορετικά επίπεδα της ταξινομίας, τότε αυτό εντάσσεται στο υψηλότερο επίπεδο που συναντάται.

Για την διαδικασία εξαγωγής συμπερασμάτων μέσω των αποτελεσματων αντλούνται επιπλέον δεδομένα από τα κωδικΌραματα. Αρχικά, υπολογίζεται για κάθε κωδικΌραμα το μέγεθος του προγράμματος, δηλαδή το άθροισμα των κελιών τα οποία δεν είναι κενά. Στη συνέχεια, σε κάθε κατηγορία της ταξινομίας SOLO υπολογίζεται ο μέσος όρος του μεγέθους αυτού. Επίσης, σε κάθε κωδικΌραμα υπολογίζεται το πλήθος των καταστάσεων του και εν συνεχεία, στο κάθε επίπεδο προσεγγίζεται ο μέσος όρος των καταστάσεων αυτών. Επιπλέον, για κάθε κωδικΌραμα εντοπίζεται το πλήθος διαδικασιών και τελικά ο αντίστοιχος μέσος όρος για κάθε επίπεδο της ταξινομίας. Τέλος, υπολογίζεται το πλήθος των κελιών με περισσότερα του ενός, παράλληλα σενάρια του ίδιου αντικειμένου στην ίδια κατάσταση και ο αντίστοιχος μέσος όρος κάθε επιπέδου.

2.2 Αποτελέσματα

Στον πίνακα 2.1 αποτυπώνονται τα αποτελέσματα κάθε ΚωδικΟράματος σχετικά με την ύπαρξη των τοπολογιών παράλληλου προγραμματισμού όπως παρουσιάζονται στην προηγούμενη ενότητα. Μετά την αξιολόγηση κάθε ΚωδικΟράματος με κριτήριο την

παραλληλία και τοποθέτησή τους στις πέντε κατηγορίες της ταξινομίας SOLO προέκυψαν οι παρακάτω συχνότητες και ποσοστά (Πίνακας 2.2 • Σχήμα 2.2.1 • Σχήμα 2.2.2).

- Στο προδομικό επίπεδο της ταξινομίας εντάσσονται ένα (1) από τα εικοσιπέντε κωδικΌραματα το οποίο αντιστοιχεί στο 4% του δείγματος.
- Στο μονοδομικό επίπεδο της ταξινομίας εντάσσονται έξι (6) από τα εικοσιπέντε κωδικΌραματα και το ποσοστό που προκύπτει είναι 24%.
- Στο πολυδομικό επίπεδο της ταξινομίας εντάσσεται και το μεγαλύτερο πλήθος κωδικΟραμάτων το οποίο είναι δεκατρία (14) από τα εικοσιπέντε και αντιστοιχεί σε ποσοστό 56% του δείγματος.
- Στο σχεσιακό επίπεδο της ταξινομίας εντάσσονται τέσσερα (4) από τα εικοσιπέντε κωδικΌραματα και το αντίστοιχο ποσοστό είναι 16%.
- Τέλος, στο επίπεδο εκτεταμένης γενίκευσης δεν εντάσσεται κανένα (0) από τα εικοσιπέντε κωδικΌραματα με αποτέλεσμα το ποσοστό να είναι 0%.

Στον πίνακα 2.3 αποτυπώνεται για κάθε ΚωδικΌραμα:

- το μέγεθος, δηλαδή το πλήθος των μη κενών κελιών
- το πλήθος των καταστάσεων
- το πλήθος διαδικασιών
- το πλήθος των κελιών με περισσότερα του ενός παράλληλα σενάρια ίδιου αντικειμένου στην ίδια κατάσταση.

Στον πίνακα 2.4 και στα σχήματα 2.2.3, 2.2.4, 2.2.5, 2.2.6 αποτυπώνονται οι μέσοι όροι των παραπάνω δεδομένων για τα κωδικΌραματα κάθε επιπέδου της SOLO.

- Στο προδομικό επίπεδο της ταξινομίας ο μέσος όρος μεγέθους ΚωδικΟράματος είναι 2, ο μέσος όρος πλήθους καταστάσεων είναι 2, ο μέσος όρος πλήθους διαδικασιών είναι 8 και ο μέσος όρος πλήθους κελιών με παράλληλα σενάρια είναι 0.
- Στο μονοδομικό επίπεδο της ταξινομίας ο μέσος όρος μεγέθους ΚωδικΟράματος είναι 17.7, ο μέσος όρος πλήθους καταστάσεων είναι 6.2, ο μέσος όρος πλήθους διαδικασιών είναι 2 και ο μέσος όρος πλήθους κελιών με παράλληλα σενάρια είναι 1.3.

- Στο πολυδομικό επίπεδο της ταξινομίας ο μέσος όρος μεγέθους ΚωδικΟράματος είναι 17.6, ο μέσος όρος πλήθους καταστάσεων είναι 8.3, ο μέσος όρος πλήθους διαδικασιών είναι 4.8 και ο μέσος όρος πλήθους κελιών με παράλληλα σενάρια είναι 0.9.
- Στο σχεσιακό επίπεδο της ταξινομίας ο μέσος όρος μεγέθους ΚωδικΟράματος είναι 19.2, ο μέσος όρος πλήθους καταστάσεων είναι 11.2, ο μέσος όρος πλήθους διαδικασιών είναι 6.2 και ο μέσος όρος πλήθους κελιών με παράλληλα σενάρια είναι 2.2.
- Τέλος, στο επίπεδο εκτεταμένης γενίκευσης δεν υπάρχουν ανάλογοι δείκτες καθώς δεν εντάσσεται κανέναν κωδικΌραμα σε αυτή.

Πίνακας 2. 1 Τοπολογίες παράλληλου προγραμματισμού στα ΚωδικΌραματα 1-25

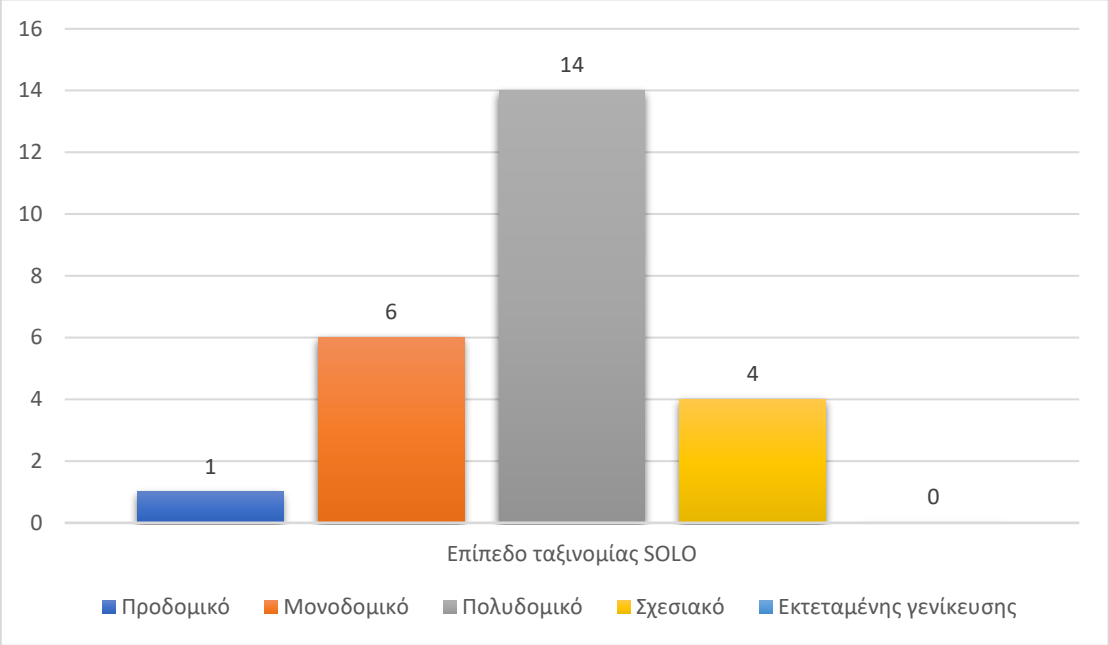
ΚωδικΌραμα	Προδομικό 1	Προδομικό 2	Μονοδομικό 1	Μονοδομικό 2	Πολυδομικό 1	Πολυδομικό 2	Σχεσιακό 1	Σχεσιακό 2	Εκτεταμένης γενίκευσης	Επίπεδο ένταξης
1			✔			✔				3
2	✔			✔						2
3			✔		✔	✔				3
4			✔					✔		4
5			✔							2
6			✔		✔	✔				3
7			✔		✔	✔				3
8			✔		✔	✔				3
9			✔		✔	✔				3
10			✔	✔		✔				3
11			✔	✔						2
12			✔		✔	✔				3
13			✔	✔	✔	✔				3
14				✔	✔	✔				3
15	✔									1
16					✔	✔				3
17			✔		✔	✔	✔			4
18				✔	✔					3
19						✔		✔		4
20			✔							2
21			✔							2
22			✔		✔					3
23			✔			✔		✔		4
24			✔	✔						2
25			✔		✔	✔				3

Παρακάτω ακολουθεί η επεξήγηση των στοιχείων που βρίσκονται στις στήλες του πίνακα 2.1.

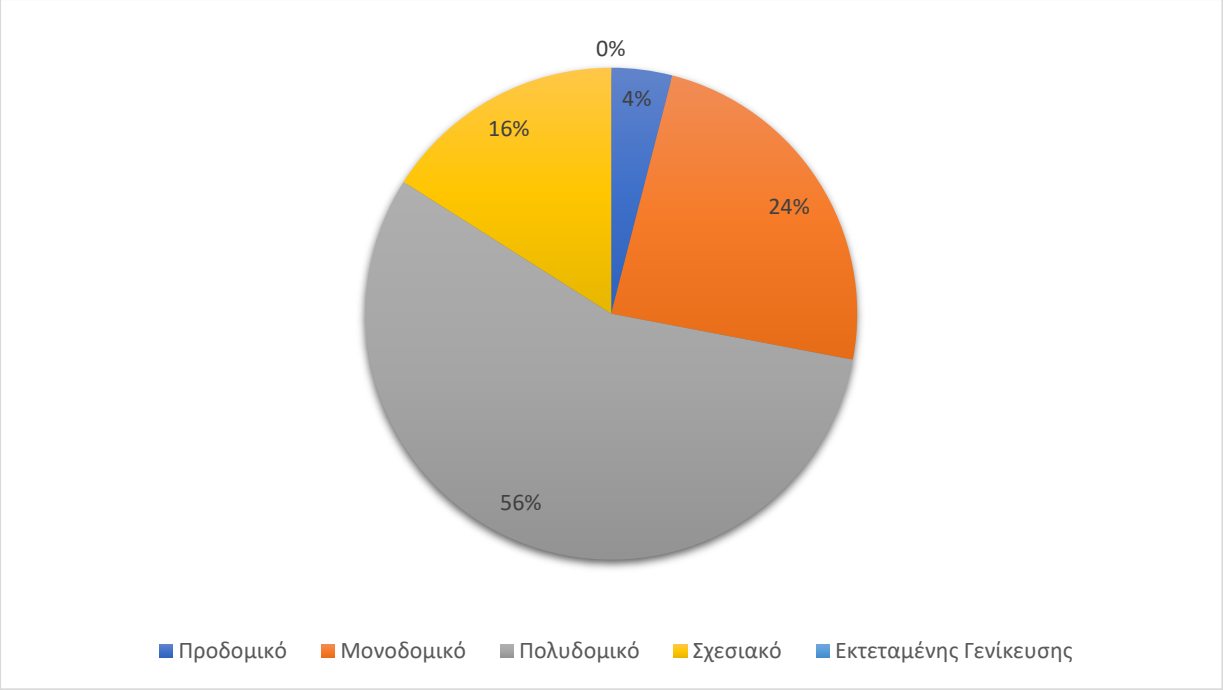
- Προδομικό 1: Σειριακός Κώδικας
- Προδομικό 2: Μη – αιτιοκρατικός κώδικας
- Μονοδομικό 1: Ταυτόχρονη εκκίνηση ανεξάρτητων, παράλληλων σεναρίων από την ίδια αιτία
- Μονοδομικό 2: Παράλληλα σενάρια που βρίσκονται σε μόνιμη κατάσταση εκτέλεσης
- Πολυδομικό 1: Μη ταυτόχρονη εκκίνηση ανεξάρτητων, παράλληλων σεναρίων από διαφορετική αιτία
- Πολυδομικό 2: Το πατρικό σενάριο εκτελείται ταυτόχρονα με τα σενάρια απογόνους που πυροδότησε
- Σχεσιακό 1: Ανάγκη επικοινωνίας των παράλληλων σεναρίων που επιτυγχάνεται με σύγχρονη επικοινωνία
- Σχεσιακό 2: Ανάγκη επικοινωνίας των παράλληλων σεναρίων που επιτυγχάνεται με ασύγχρονη επικοινωνία
- Εκτεταμένης σχεδίασης: Πρόσβαση σε κρίσιμες περιοχές κώδικα από τα παράλληλα σενάρια με τη χρήση σηματωρών
- Επίπεδο ένταξης: Επίπεδο ένταξης του προγράμματος στην ταξινομία SOLO

Πίνακας 2. 2 Συχνότητα και ποσοστό κωδικΟραμάτων για κάθε κατηγορία της SOLO

Επίπεδο	Συχνότητα	Ποσοστό
Προδομικό	1	4%
Μονοδομικό	6	24%
Πολυδομικό	14	56%
Σχεσιακό	4	16%
Εκτεταμένης γενίκευσης	0	0%



Σχήμα 2.2. 1 Συχνότητα κωδικΟραμάτων για κάθε κατηγορία της SOLO



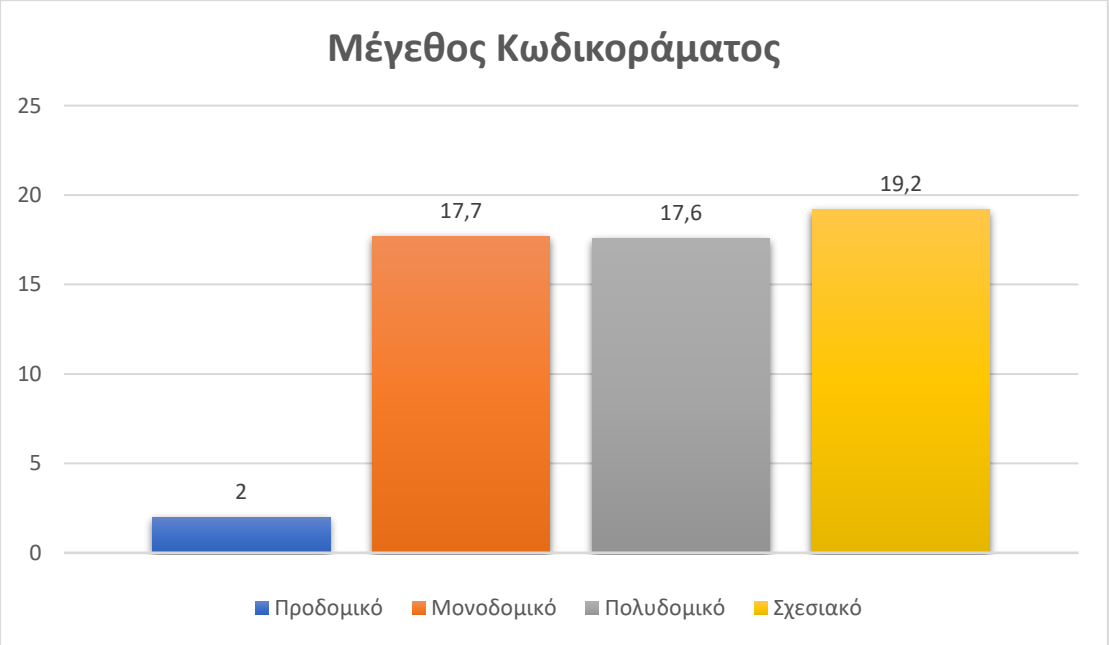
Σχήμα 2.2. 2 Ποσοστό κωδικΟραμάτων για κάθε κατηγορία της SOLO

Πίνακας 2. 3 Δεδομένα για το μέγεθος, το πλήθος καταστάσεων, το πλήθος διαδικασιών και το πλήθος κελιών με παράλληλα σενάρια του ίδιου αντικειμένου για τα κωδικΌραματα 1 έως 25

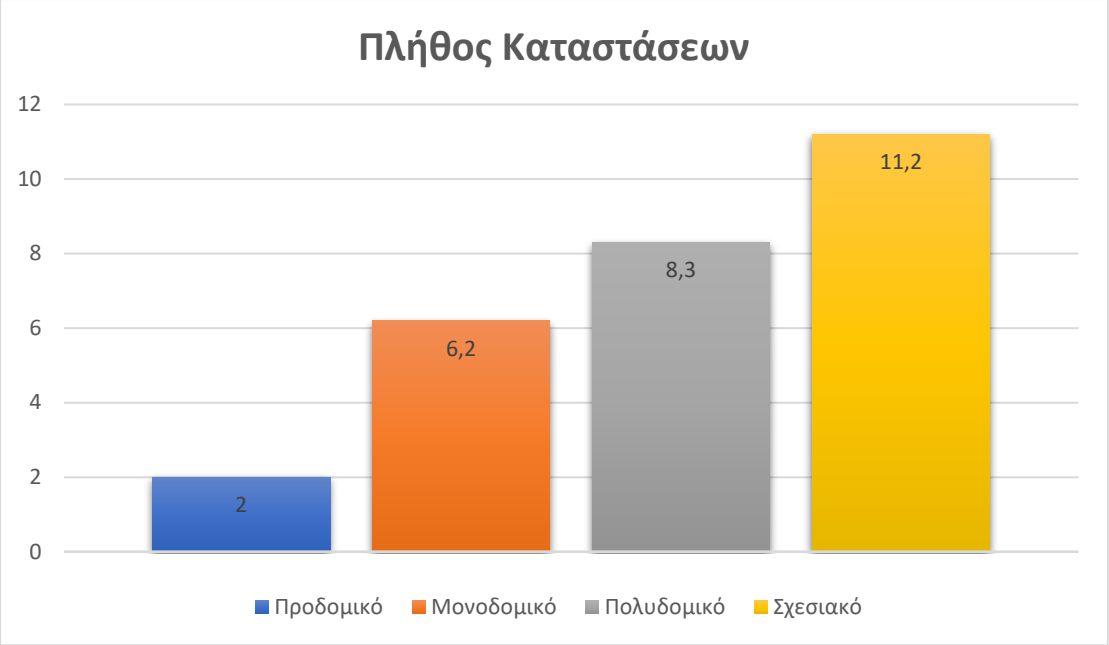
ΚωδικΌραμα	Μέγεθος ΚωδικΟράματος (Πλήθος μη κενών κελιών)	Πλήθος καταστάσεων	Πλήθος διαδικασιών	Πλήθος κελιών με παράλληλα σενάρια του ίδιου αντικειμένου
1	20	8	0	4
2	10	6	0	0
3	45	16	6	2
4	16	5	0	9
5	21	8	0	3
6	21	8	10	0
7	14	9	3	0
8	13	5	4	0
9	13	7	0	0
10	10	6	0	2
11	13	3	0	0
12	18	6	0	1
13	19	16	0	0
14	9	6	3	0
15	2	2	8	0
16	8	6	4	1
17	18	13	3	0
18	11	7	33	1
19	25	21	12	0
20	24	8	6	0
21	19	5	6	0
22	26	6	4	0
23	18	6	10	0
24	19	7	0	5
25	19	10	0	2

Πίνακας 2. 4 Μέσος όρος για το μέγεθος, το πλήθος καταστάσεων, το πλήθος διαδικασιών και το πλήθος κελιών με παράλληλα σενάρια του ίδιου αντικειμένου, των κωδικΟραμάτων κάθε επιπέδου της ταξινομίας SOLO

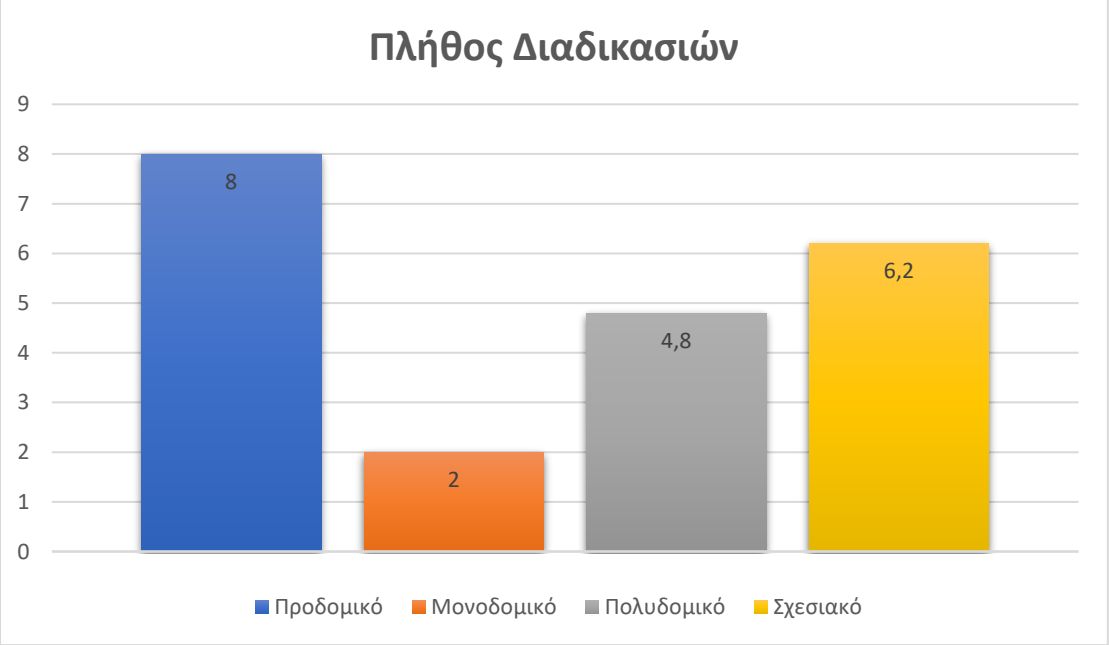
Δεδομένο	Προδομικό	Μονοδομικό	Πολυδομικό	Σχεσιακό	Εκτεταμένης Γενίκευσης
Μέγεθος ΚωδικΟράματος (Πλήθος μη κενών κελιών)	2	17.7	17.6	19.2	-
Πλήθος καταστάσεων	2	6.2	8.3	11.2	-
Πλήθος διαδικασιών	8	2	4.8	6.2	-
Πλήθος κελιών με παράλληλα σενάρια του ίδιου αντικειμένου	0	1.3	0.9	2.2	-



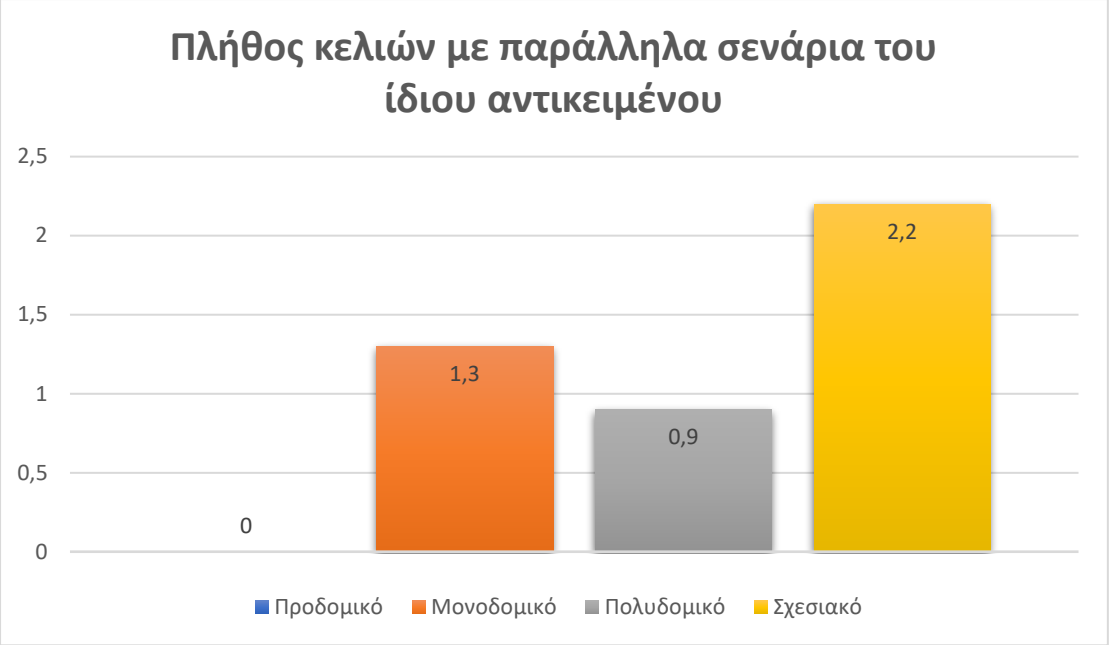
Σχήμα 2.2. 3 Μέσος όρος για το μέγεθος των κωδικΟραμάτων κάθε επιπέδου της ταξινομίας SOLO



Σχήμα 2.2. 4 Μέσος όρος για το πλήθος των καταστάσεων των κωδικΟραμάτων κάθε επιπέδου της ταξινομίας SOLO



Σχήμα 2.2. 5 Μέσος όρος για το πλήθος των διαδικασιών των κωδικΟραμάτων κάθε επιπέδου της ταξινομίας SOLO



Σχήμα 2.2. 6 Μέσος όρος για το πλήθος κελιών με παράλληλα σενάρια του ίδιου αντικειμένου των κωδικΟραμάτων κάθε επιπέδου της ταξινομίας SOLO

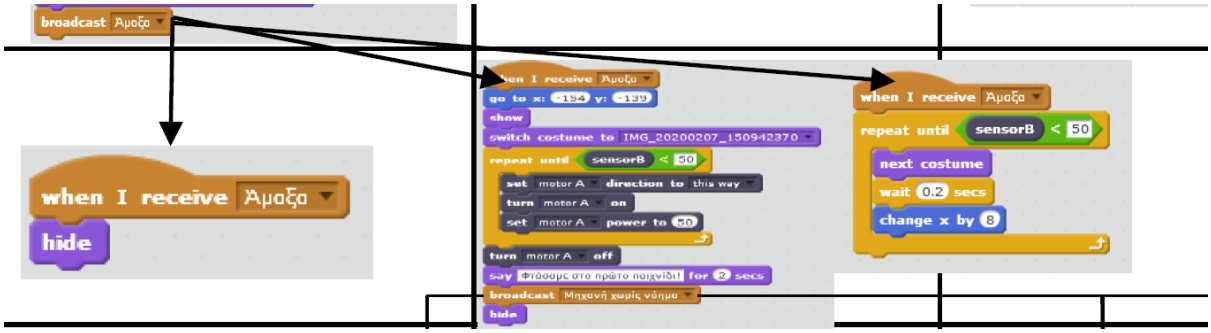
2.3 Συζήτηση και συμπεράσματα

Σκοπός της παρούσας εργασίας είναι να διερευνηθεί κατά πόσο οι μαθητές δέκα έως δεκατριών ετών μπορούν να σκεφτούν «παράλληλα» και να χρησιμοποιήσουν αυτές τις διεργασίες της υπολογιστικής τους σκέψης στον προγραμματιστικό κώδικα τους. Για την διερεύνηση αυτή, παρακάτω συζητούνται τα αποτελέσματα που προέκυψαν από την ένταξη των κωδικΟραμάτων στα επίπεδα της ταξινομίας SOLO με κριτήριο την παραλληλία. Επίσης, συζητούνται οι τοπολογίες παράλληλου προγραμματισμού που χρησιμοποιούν οι μαθητές σε κάθε κατηγορία και τα διάφορα στατιστικά στοιχεία που προκύπτουν για κάθε επίπεδο.

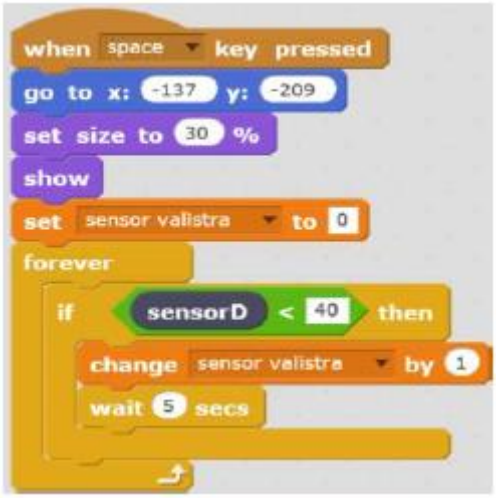
Από τα σχήματα 2.2.1 και 2.2.2 και τον πίνακα 2.2 προκύπτει το ποσοστό των κωδικΟραμάτων που ανήκουν στο προδομικό επίπεδο της ταξινομίας SOLO και είναι μόλις 4%. Ένα τόσο χαμηλό ποσοστό σε αυτή την κατηγορία, δείχνει πως σχεδόν όλοι οι μαθητές δεν χρησιμοποιούν αποκλειστικά σειριακούς κώδικες και πως δεν έχουν περιορισμένη αντίληψη για την διαχείριση του κώδικά τους ως προς την παραλληλία.

Στο μονοδομικό επίπεδο της ταξινομίας προκύπτει ένα ποσοστό 24%. Η ερμηνεία αυτού του ποσοστού είναι πως σε ένα στα τέσσερα κωδικΌραματα παρατηρείται μια περιορισμένη

χρήση τοπολογιών παράλληλου προγραμματισμού χωρίς μεταξύ τους συνδέσεις, καθώς λειτουργούν ανεξάρτητα και ξεκινούν ταυτόχρονα. Ένα χαρακτηριστικό παράδειγμα αποτελεί το κωδικΌραμα 24 το οποίο εντάσσεται στο μονοδομικό επίπεδο. Στο σχήμα 2.3.1 αποτυπώνεται ένα στιγμιότυπο του κώδικα του ΚωδικΟράματος αυτού, κατά το οποίο η ίδια αιτία πυροδοτεί την ταυτόχρονη εκκίνηση παράλληλων σεναρίων. Επίσης, όπως δείχνουν ο πίνακας 2.1 σε αυτό το επίπεδο οι μαθητές χρησιμοποιούν παράλληλα σενάρια που βρίσκονται σε μια διαρκή επανάληψη έως ότου ικανοποιηθεί κάποια συνθήκη. Στο σχήμα 2.3.2 αποτυπώνεται ένα στιγμιότυπο του ΚωδικΌραματος 2 που δείχνει μια τέτοια ατέρμονη επανάληψη. Η επανάληψη αυτή βρίσκεται σε κατάσταση εκτέλεσης, διαρκεί για πάντα και ελέγχει αν ισχύει κάποια συνθήκη για να εκτελέσει ένα τμήμα κώδικα. Το Scratch δίνει την δυνατότητα να χρησιμοποιηθεί η εντολή "Όταν" ώστε το σενάριο να βρίσκεται σε κατάσταση επαγρύπνησης αλλά στις περιπτώσεις αυτές οι μαθητές δεν το χρησιμοποιούν.

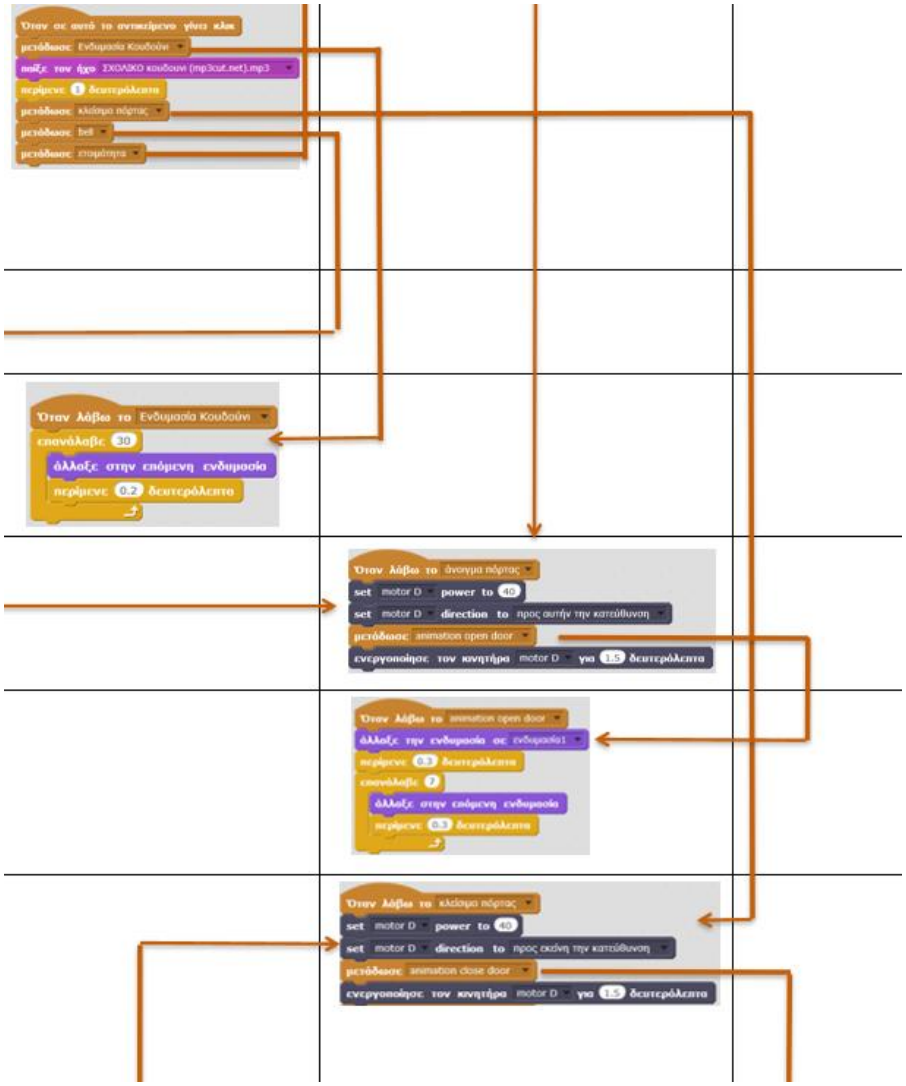


Σχήμα 2.3. 1 Στιγμιότυπο του ΚωδικΟράματος 24 που η ίδια αιτία πυροδοτεί την ταυτόχρονη εκκίνηση παράλληλων σεναρίων

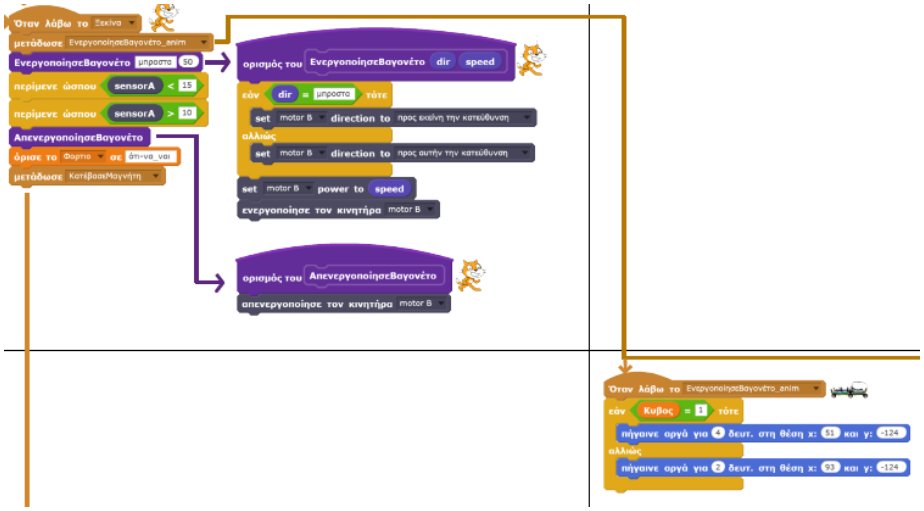


Σχήμα 2.3. 2 Στιγμιότυπο του ΚωδικΟράματος 2 που δείχνει ένα τμήμα κώδικα σε μόνιμη κατάσταση επανάληψης

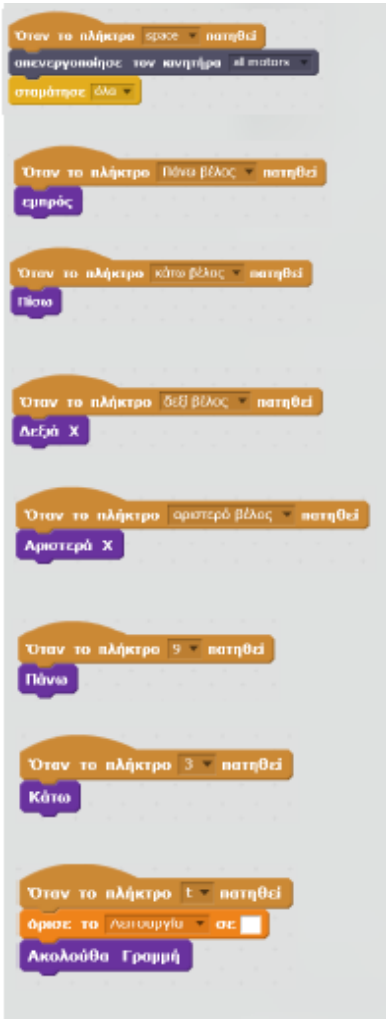
Στο πολυδομικό επίπεδο βρίσκεται η πλειοψηφία των κωδικΟραμάτων με ένα ποσοστό 56%. Αυτό το ποσοστό δείχνει πως στην υπολογιστική σκέψη των περισσοτέρων μαθητών υπάρχουν οι διεργασίες για την χρήση ποικίλων παράλληλων σεναρίων με διάφορους τρόπους, αλλά χωρίς να υπάρχουν σημαντικές συνδέσεις και επικοινωνία μεταξύ τους. Σε αυτή την κατηγόρια το βασικότερο χαρακτηριστικό είναι πως υπάρχουν παράλληλα σενάρια που δεν ξεκινούν ταυτόχρονα καθώς δεν πυροδοτούνται από την ίδια αιτία. Τέτοιο παράδειγμα αποτελεί το στιγμιότυπο του ΚωδικΟράματος 13 στο σχήμα 2.2.3. Ανάλογο παράδειγμά αποτυπώνεται στο σχήμα 2.2.4 του ΚωδικΟράματος 6 που το πατρικό σενάριο συνεχίζει να εκτελείται παράλληλα με το σενάριο απόγονο κάτι το οποίο συμβαίνει συχνά σύμφωνα με τον πίνακα 2.1. Επίσης, από το ποσοστό αυτού του επίπεδου φαίνεται πως οι μαθητές στην υπολογιστική τους σκέψη χρησιμοποιούν το “όταν” και όχι το “αν” σε μια παντοτινή επανάληψη, κάτι που πιθανότατα προκύπτει από την καθημερινότητά τους και τον τρόπο σκέψης τους σε αυτή. Χαρακτηριστικό παράδειγμα με την χρήση του “όταν” αποτελεί το κωδικΌραμα 18 του οποίου το στιγμιότυπο αποτυπώνεται στο σχήμα 2.2.5. Όμως, σε πολλές περιπτώσεις που υπήρχε η ανάγκη συγχρονισμού των σεναρίων, αυτή επιτυγχανόταν με ανορθόδοξο τρόπο, δηλαδή με την εντολή “Περίμενε x δευτερόλεπτα”, όπου x ένας πραγματικός αριθμός, ώστε να ολοκληρωθεί κάτι πριν συνεχίσει ο κώδικας. Ο συγχρονισμός δεν πραγματοποιούταν με τη χρήση μεταβλητών ή τη χρήση της εντολής “Μετέδωσε και περίμενε”, δηλαδή τεχνικές παράλληλου προγραμματισμού που θα κατηγοριοποίησαν το κωδικΌραμα σε κάποιο παραπάνω επίπεδο της SOLO. Στο σχήμα 2.3.6 αποτυπώνεται μια τέτοια περίπτωση του ΚωδικΟράματος 3 που ο κώδικας χρησιμοποιεί στο πατρικό σενάριο την εντολή “περίμενε 0.5 δευτερόλεπτα” μετά την πυροδότηση ενός άλλου σεναρίου ώστε να μην τρέξει παράλληλα με αυτό.



Σχήμα 2.3. 3 Στιγμιότυπο του ΚωδικΟράματος 13 που διαφορετική αιτία πυροδοτεί την εκκίνηση παράλληλων σεναρίων



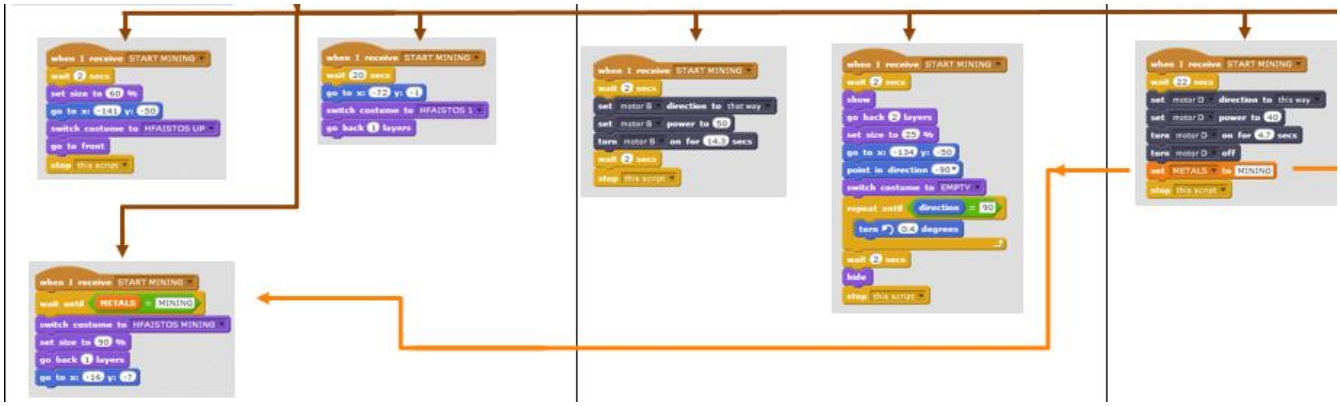
Σχήμα 2.3. 4 Στιγμιότυπο του ΚωδικΟράματος 6 κατά το οποίο το πατρικό σενάριο συνεχίζει να εκτελείται παράλληλα με το σενάριο απόγονο



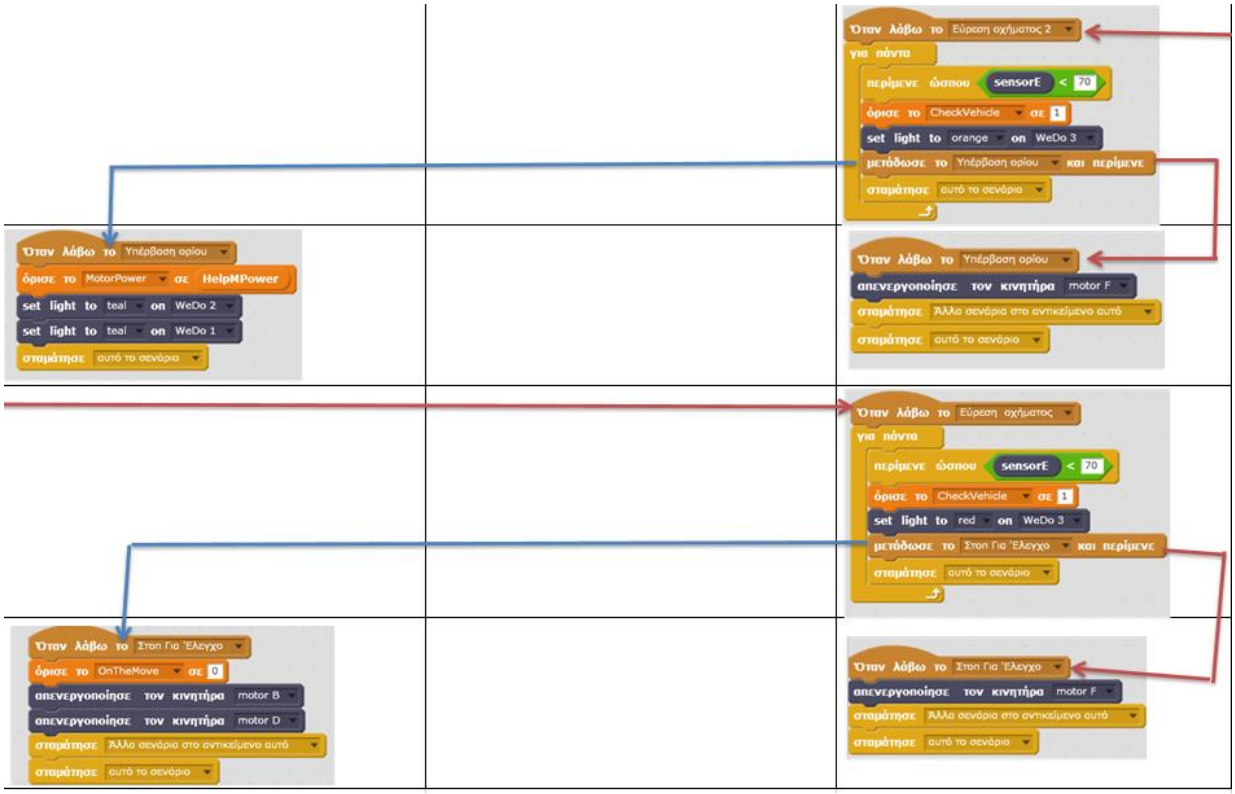
Σχήμα 2.3. 5 Στιγμιότυπο του ΚωδικΟράματος 18 που γίνεται χρήση της εντολής "Όταν" και όχι της "Αν"

Σχήμα 2.3. 6 Στιγμιότυπο του ΚωδικΟράματος 3 που γίνεται χρήση της εντολής “Περίμενε x δευτερόλεπτα”

Από τα αποτελέσματα προκύπτει πως το ποσοστό κωδικΟραμάτων που ανήκουν στο σχεσιακό επίπεδο της SOLO είναι 16%. Αυτό το νούμερο υποδηλώνει πως κάποιοι μαθητές παρά τη νεαρή τους ηλικία εκτιμούν την αναγκαιότητα επικοινωνίας των διαφόρων σεναρίων και την επιτυγχάνουν είτε σύγχρονα, είτε ασύγχρονα. Στο σχήμα 2.3.7 αποτυπώνεται ένα τμήμα του ΚωδικΟράματος 4 το οποίο εντάσσεται στο σχεσιακό επίπεδο. Στο συγκεκριμένο στιγμιότυπο επιτυγχάνεται ασύγχρονη επικοινωνία, καθώς ένα τμήμα κώδικα περιμένει να αλλάξει τιμή η μεταβλητή “METALS”. Αυτή η αλλαγή μεταβλητής πραγματοποιείται σε κάποιο σενάριο και ακολουθεί η ενημέρωση στο σενάριο που αναμένει. Στο σχήμα 2.3.8 αποτυπώνεται ένα στιγμιότυπο του ΚωδικΟράματος 17 στο οποίο επιτυγχάνεται μια μορφή σύγχρονης επικοινωνίας. Αυτό πραγματοποιείται με τη χρήση της εντολής “Μετέδωσε και Περίμενε” με αποτέλεσμα το πατρικό σενάριο να αναμένει το σενάριο απόγονο να ολοκληρωθεί, ώστε να συνεχίσει τη λειτουργία του. Συμπερασματικά, στους κώδικες αυτούς γίνεται αντιληπτή η σημασία των διαφόρων τμημάτων σε σχέση με το σύνολο του ΚωδικΟράματος.



Σχήμα 2.3. 7 Στιγμιότυπο του ΚωδικΟράματος 4 που επιτυγχάνεται ασύγχρονη επικοινωνία με τη χρήση μεταβλητής



Σχήμα 2.3. 8 Στιγμιότυπο του ΚωδικΟράματος 17 που επιτυγχάνεται σύγχρονη επικοινωνία με χρήση της εντολής “Μετέδωσε και Περίμενε”

Στο τελευταίο επίπεδο της SOLO, αυτό της εκτεταμένης γενίκευσης, το ποσοστό των κωδικΟραμάτων είναι 0%. Αυτό σημαίνει πως οι μαθητές δεν επιλύουν προβλήματα πρόσβασης σε κρίσιμη περιοχή κώδικα διαχείρισης κοινόχρηστου πόρου.

Τα αποτελέσματα του σχήματος 2.2.3 αναδεικνύουν τον μέσο όρο του μεγέθους (πλήθους κελιών) των κωδικΟραμάτων για κάθε επίπεδο της ταξινομίας SOLO. Μεταξύ του προδομικού επιπέδου και των άλλων επιπέδων της SOLO υπάρχει μια μεγάλη διαφορά μεγέθους. Όμως μεταξύ μονοδομικού, πολυδομικού και σχεσιακού επιπέδου δεν προκύπτει κάποια διαφορά. Στο προδομικό επίπεδο εντάσσεται μόλις ένα κωδικΌραμα το οποίο έχει έναν πολύ μικρό αριθμό μη κενών κελιών καθώς όλος ο κώδικας βρίσκεται συμπυκνωμένος σε μόλις δύο κελιά. Ένα τόσο μικρός πλήθος κελιών με σειριακό κώδικα δεν επιτρέπει την παράλληλη λειτουργία ποικίλων σεναρίων και επικρατεί ο δομημένος προγραμματισμός.

Από το σχήμα 2.2.4 προκύπτει πως σε όσο μεγαλύτερη κατηγορία της ταξινομίας SOLO ανήκει ένα κωδικΌραμα τόσο μεγαλύτερο πλήθος καταστάσεων έχει. Αυτό το αποτέλεσμα αναδεικνύει την πιθανότητα της συνύπαρξης πολλών καταστάσεων με τοπολογίες παράλληλου προγραμματισμού που αντιστοιχούν σε υψηλά επίπεδα της ταξινομίας. Φαίνεται πως στους κώδικες των μαθητών με μεγάλο πλήθος καταστάσεων υπάρχει και μεγαλύτερη ανάγκη για παραλληλία μεταξύ των σεναρίων. Αυτή η ανάγκη γίνεται αντιληπτή από τους μαθητές και γίνεται χρήση ανάλογων τοπολογιών παράλληλου προγραμματισμού.

Το σχήμα 2.2.5 δείχνει τον μέσο όρο του πλήθους των διαδικασιών των κωδικΟραμάτων για κάθε επιπέδου της ταξινομίας SOLO. Τα αποτελέσματα δείχνουν πως στο προδομικό επίπεδο γίνεται χρήση διαδικασιών σε μεγάλο βαθμό σε σχέση με τα άλλα επίπεδα της ταξινομίας. Αυτό το γεγονός αναδεικνύει πως εκεί που δεν γίνεται χρήση τοπολογιών παράλληλου προγραμματισμού, επικρατούν στοιχεία συναρτησιακού προγραμματισμού όπως είναι η ύπαρξη και η χρήση πολλαπλών διαδικασιών. Στις υπόλοιπες κατηγορίες (μονοδομικό, πολυδομικό, σχεσιακό) της ταξινομίας φαίνεται να υπάρχει αύξηση του πλήθους των διαδικασιών όσο αυξάνεται το επίπεδο ένταξης του κώδικα. Κάτι τέτοιο ίσως συμβαίνει καθώς στα μεγαλύτερα επίπεδα η πολυπλοκότητα αυξάνεται και η χρήση διαδικασιών εξυπηρετεί μια τέτοια ανάγκη.

Τέλος, το σχήμα 2.2.6. αφορά τον μέσο όρο του πλήθους των κελιών με παράλληλα σενάρια του ίδιου αντικειμένου των κωδικΟραμάτων κάθε επιπέδου της ταξινομίας SOLO. Από το

αποτέλεσμα αυτό δεν προκύπτει κάποια παρατήρηση καθώς οι διαφορές μεταξύ των επιπέδων είναι πολύ μικρές.

Το βασικό συμπέρασμα που εξάγεται από την ερμηνεία των αποτελεσμάτων, είναι πως οι μαθητές μπορούν ακόμη και από νεαρή ηλικία να σκεφτούν παράλληλα και να χρησιμοποιήσουν με επιτυχία στον κώδικα τους τον παράλληλο προγραμματισμό. Το παραπάνω συμφωνεί με τον Kahn (1996) ο οποίος τονίζει ότι ο παράλληλος προγραμματισμός είναι πιο κοντά στο φυσικό τρόπο σκέψης των νεαρών μαθητών καθώς στον πραγματικό κόσμο πολλαπλές οντότητες δρουν σε αυτόν ταυτόχρονα και επικοινωνούν μεταξύ τους. Τα αποτελέσματα της έρευνας δείχνουν πως οι μαθητές δέκα έως δεκατριών ετών χρησιμοποιούν τοπολογίες παράλληλου προγραμματισμού όταν τους δίνεται αυτή η δυνατότητα, όπως δηλαδή συμβαίνει στο περιβάλλον οπτικού προγραμματισμού Scratch. Η βιβλιογραφική επισκόπηση συμφωνεί με αυτό καθώς οι μαθητές ηλικιών δημοτικού σε ποικίλες μελέτες αναδείχθηκε πως μετά από τη διαδικασία διδασκαλίας άρχισαν να σκέφτονται παράλληλα και να χρησιμοποιούν παράλληλο κώδικα με επιτυχία (Gregg et al., 2012· Sáez-López et al., 2016· Fatourou et al., 2018· Weintrop et al., 2018), αλλά και σε μεγαλύτερες ηλικίες γυμνασίου - λυκείου (Feldhausen et al., 2014 · Pellas, 2014). Σύμφωνα με τους Weintrop et al. το 2018, ακόμα και δεν έχουν διδαχθεί στοιχεία παραλληλίας οι μαθητές παράγουν παράλληλα τμήματα κώδικα χωρίς να το καταλαβαίνουν, κάτι που επιβεβαιώνεται και από την παρούσα έρευνα καθώς επίπεδο διδασκαλίας πάνω στον παράλληλο προγραμματισμό που προηγήθηκε της δημιουργίας κωδικΟραμάτων είναι μικρό. Επίσης, γίνεται αντιληπτό ότι αρκετές φορές προτιμούν να χρησιμοποιούν το “Όταν” ή στοιχεία που παραπέμπουν σε αυτό και όχι το “Για πάντα… Αν”. Κάτι τέτοιο έρχεται σε συμφωνία με τους Tarkan et al. οι οποίοι το 2010 συμπέραναν πως οι μαθητές προτιμούν τις παράλληλες διεργασίες σε σχέση με τη δομή επανάληψης. Πιο συγκεκριμένα, προτιμούν πολλαπλές οντότητες να λειτουργούν παράλληλα εκτελώντας διεργασίες παρά μία οντότητα να εκτελεί όλες τις διεργασίες μέσω της δομής επανάληψης. Επιπλέον, από την παρούσα έρευνα δεν μπορεί να αξιολογηθεί η προϋπάρχουσα αντίληψη των μαθητών για το τι είναι παράλληλος προγραμματισμός καθώς δεν υπάρχουν τέτοια στοιχεία, αλλά από την βιβλιογραφία προκύπτει πως οι μαθητές δεν ξέρουν τι είναι χωρίς να έχουν διδαχθεί για αυτόν (Libert & Vanhoof, 2019). Τέλος, όσον αφορά την πλατφόρμα οπτικού προγραμματισμού Scratch, με βάση τη μεγάλη χρήση παράλληλου προγραμματισμού από

τους μαθητές που παρατηρήθηκε στην παρούσα εργασία, αναδεικνύεται πως προσφέρεται για την διδασκαλία παράλληλου προγραμματισμού σε αρχάριους. Το συμπερασμα αυτό έρχεται σε συμφωνία με τους Λαδιά και Καρβουνίδη που το 2021 τονίζουν τη χρησιμότητα αυτή του Scratch. Επίσης, η πλειοψηφία των άλλων ερευνών, που χρησιμοποιήσαν αυτή την πλατφόρμα οπτικού προγραμματισμού είχαν ανάλογα αποτελέσματα (Feldhausen et al., 2014 · Pellas, 2014· Sáez-López et al., 2016· Fatourou et al., 2018 · Weintrop et al., 2018).

2.3.1. Περιορισμοί της έρευνας

Ο βασικός περιορισμός της έρευνας είναι το μικρό δείγμα που χρησιμοποιήθηκε καθώς αναλύθηκαν μόλις είκοσι πέντε κωδικΟράματα από τον Πανελλήνιο Διαγωνισμό Εκπαιδευτικής Ρομποτικής του WRO – Hellas. Τα κωδικΟράματα που χρησιμοποιήθηκαν είναι αυτά που είχαν ξεχωρίσει στο διαγωνισμό. Επιπλέον, περιορισμό αποτελεί η χρήση κωδικΟραμάτων μόνο από μια συγκεκριμένη χρόνια, και ειδικότερα το 2020. Επίσης, οι εκπαιδευτικοί που καθοδήγησαν τις ομάδες των μαθητών κατά την παραγωγή κώδικα έχουν χαμηλό επίπεδο γνώσης στοιχείων παράλληλου προγραμματισμού, χωρίς όμως αυτό το επίπεδο να είναι απόλυτα γνωστό.

2.3.2 Προτάσεις για μελλοντική έρευνα

Για μελλοντική έρευνα προτείνεται η πραγματοποίηση της μελέτης με μεγαλύτερο πλήθος προγραμμάτων για δείγμα και σε μεγαλύτερο εύρος ηλικίας μαθητών από διαφορετικές χρόνιες του διαγωνισμού ρομποτικής, ώστε τα αποτελέσματα να είναι πιο ακριβή και να μπορούν να γενικευτούν. Επιπλέον, προτείνεται να σχεδιαστεί μια έρευνα ώστε να αναδειχθεί το υπάρχον γνωστικό επίπεδο των εκπαιδευτικών στις τοπολογίες παράλληλου προγραμματισμού στο Scratch αλλά και πάνω στον παράλληλο προγραμματισμό γενικότερα.

Αναφορές

Δημακόπουλος Β., (2015). *Παράλληλα Συστήματα και Προγραμματισμός.* [ηλεκτρ. βιβλ.] Αθήνα: Σύνδεσμος Ελληνικών Ακαδημαϊκών Βιβλιοθηκών. Ανακτήθηκε Ιανουάριο 15, 2021, από : http://hdl.handle.net/11419/3209

Λαδιάς, Δ., Καρβουνίδης, Θ., Λαδιάς, Α. (2020). Αξιοποίηση τεχνικών παράλληλου προγραμματισμού στο περιβάλλον Scratch. *12th CIE2020 - Η Πληροφορική στην Εκπαίδευση* (σ. 2-13). Πειραιάς.

Λαδιάς, Δ. & Καρβουνίδης, Θ. (2021). Τοπολογίες παράλληλου προγραμματισμού στο Scratch και κατηγοριοποίησή τους με χρήση της ταξινομίας SOLO. *13th CIE2020 - Η Πληροφορική στην Εκπαίδευση* (σ. 484-499). Κέρκυρα.

Λαδιάς, Α. & Λαδιάς Δ. (2016). Αναπαράσταση αλγορίθμων με τη βοήθεια του ΚωδικΟράματος σε περιβάλλοντα οπτικού προγραμματισμού. *Θέματα Επιστημών και Τεχνολογίας στην Εκπαίδευση*, *9*(2), 103-117.

Μαστοροκώστας Π. (2015). *Διαδικαστικός Προγραμματισμός.* [ηλεκτρ. βιβλ.] Αθήνα: Σύνδεσμος Ελληνικών Ακαδημαϊκών Βιβλιοθηκών. Ανακτήθηκε Φεβρουάριο 18, 2021 από: https://repository.kallipos.gr/bitstream/11419/1346/1/00_master%20document_ΚΟΥ.pdf

Σακελλαρίου, Η., Βασιλειάδης, Ν., Κεφαλάς, Π., Σταμάτης, Δ., (2015). *Τεχνικές λογικού προγραμματισμού.* [ηλεκτρ. βιβλ.] Αθήνα: Σύνδεσμος Ελληνικών Ακαδημαϊκών Βιβλιοθηκών. Ανακτήθηκε Φεβρουάριο 17, 2021 από: http://hdl.handle.net/11419/777

Σταματόπουλος, Π. (2015). *Συναρτησιακός Προγραμματισμός - Υπολογισμός με Συναρτήσεις.* [ηλεκτρ. βιβλ.] Αθήνα: Σύνδεσμος Ελληνικών Ακαδημαϊκών Βιβλιοθηκών. κεφ. 9. Ανακτήθηκε Φεβρουάριο 17, 2021 από: http://hdl.handle.net/11419/3595

Aho, A. V. (2012). Computation and Computational Thinking. *The Computer Journal, 55*(7), 832–835.

Avacheva, T. & Prutzkow, A. (2020). The Evolution of Imperative Programming Paradigms as a Search for New Ways to Reduce Code Duplication*. IOP Conference Series: Materials Science and Engineering, 714*(1).

Bal, H. E., Steiner, J. G., & Tanenbaum, A. S. (1989) Programming Languages for Distributed Computing Systems. *ACM Computing Surveys, 21*(3), 261-322.

Barr, V. & Stephenson, C. (2011). Bringing computational thinking to K-12: what is Involved and what is the role of the computer science education community?. *ACM Inroads, 2*(1), 48-54.

Biggs, J. B., & Collis, K. F. (1982). *Evaluating the quality of learning. The SOLO taxonomy.* NY: Academic Press.

Bustard, D. W. (1990). *Concepts of Concurrent Programming.* SEI-CM-24, Carnegie Mellon University: Software Engineering Institute.

Fatourou, E., Zygouris, N. C., Loukopoulos, T., & Stamoulis, G. I. (2018). Teaching concurrent programming concepts using scratch in primary school: Methodology and evaluation. *International Journal of Engineering Pedagogy, 8*(4), 543–552.

Feldhausen, R., Bell, S., Andresen, D. (2014). Minimum Time, Maximum Effect: Introducing Parallel Computing in CS0 and STEM Outreach Activities Using Scratch. *Conference: Proceedings of the Extreme Science and Engineering Discovery Environment* (pp. 1-7). Atlanta, GA.

Gregg, C., Tychonievich, L., Cohoon, J., & Hazelwood, K. (2012). EcoSim: A Language and Experience Teaching Parallel Programming in Elementary School. *Proceedings of the 43rd ACM Technical Symposium on Computer Science Education (SIGCSE '12)* (pp. 51-56). NC: ACM.

Haberman, B. & Kolikant, Y.B.D. (2001), Activating «Black Boxes» instead of opening «Zippers» - a method of teaching novices basic CS concepts. *Proceedings of the ACM ITiCSE '01 Conference* (pp. 41-44). Canterbury, UK: ACM.

Hughes, J. (1989). Why Functional Programming Matters. *The Computer Journal, 32*(2), 98–107.

Hunt, K. P. (1979). An introduction to structured programming. *Behavior Research Methods & Instrumentation, 11*(2), 229-233.

Jain, L. C. & Nguyen, N. T. (Eds.). (2009). *Knowledge Processing and Decision Making in Agent-Based Systems.* Studies in Computational Intelligence.

Kahn, K. (1996). Drawing on napkins, video-game animation, and other ways to program computers. *Communications of the ACM, 39*(8), 49-59.

Kalelioğlu, F. (2015). A new way of teaching programming skills to K-12 students: Code.org. *Computers in Human Behavior, 52*, 200–210.

Keren, G., & Fridin, M. (2014). Kindergarten Social Assistive Robot (KindSAR) for children's geometric thinking and metacognitive development in preschool education: A pilot study. *Computers in Human Behavior, 35*, 400–412.

Kirk, D. B. & Hwu, W. W. (2006). *Programming Massively Parallel Processors.* Cambridge, MA, United States: Elsevier.

Krippendorff K. (2004). *Content analysis: An introduction to its methodology.* Thousand Oaks, CA: Sage Publications.

Lahtinen, E., Ala-Mutka, K., & Järvinen, H.-M. (2005). A study of the difficulties of novice programmers. *ACM SIGCSE Bulletin, 37*(3), 14-18.

Lee, I., Martin, F., Denner, J., Coulter, B., Allan, W., Erickson, J., Malyn-Smith, J., & Werner, L. (2011). Computational thinking for youth in practice. *ACM Inroads, 2*(1), 32-37.

Libert, C., Vanhoof, W. (2019). Analysis of students' preconceptions of concurrency. *EASEAI 2019: Proceedings of the 1st ACM SIGSOFT International Workshop on Education through Advanced Software Engineering and Artificial Intelligence* (pp. 9-12). Tallin: ACM.

Lidtke, D.K. & Zhou, H.H. (1999). A new approach to an introduction to Computer Science. *Proceedings of the 29th ASEE/IEEE Frontiers in Education Conference* (pp. 12-23). Puerto Rico: IEEE.

Lischner, R. (2001), Explorations: Structured Labs for First-Time Programmers. *Proceedings of the ACM SIGCSE '01 Conference* (pp. 154-158). Charlotte, USA: ACM.

Liu, C. C., Cheng, Y. B., & Huang, C. W. (2011). The effect of simulation games on the learning of computational problem solving. *Computers & Education, 57*, 1907–1918.

Maloney, J., Resnick, M., Rusk, N., Silverman, B., & Eastmond, E. (2010). The Scratch Programming Language and Environment. *ACM Transactions on Computing Education, 10*(4), 1–15.

Michaelson, G. (2015). Teaching Programming with Computational and Informational Thinking. *Journal of Pedagogic Development, 5*(1).

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B., & Kafai, Y. (2009). Scratch: Programming for All. *Communications of the ACM, 52*(11), 60-67.

Rich, P.J., Browning, S.F., Perkins, M. et al. (2019). Coding in K-8: International Trends in Teaching Elementary/Primary Computing. *TechTrends 63*(3), 311–329.

Robins, A., Rountree, J., & Rountree, N. (2003). Learning and Teaching Programming: A Review and Discussion. *Computer Science Education, 13*(2), 137–172.

Pellas, N. (2014). The development of a virtual learning platform for teaching concurrent programming languages in Secondary education: The use of Open Sim and Scratch4OS. *Journal of E-Learning and Knowledge Society 10*(1).

Sáez-López, J.-M., Román-González, M., & Vázquez-Cano, E. (2016). Visual programming languages integrated across the curriculum in elementary school: A two-year case study using "Scratch" in five schools. *Computers & Education, 97*, 129–141.

Samuel, M. (2017). An insight to Programming Paradigms and Their Programming Languages. *Journal of Applied Technology and Innovation, 1*(1), 37-57.

Shute, V. J., Sun, C., & Asbell-Clarke, J. (2017). Demystifying computational thinking. *Educational Research Review, 22*, 142–158.

Tarkan, S., Sazawal, V., Druin, A., Golub, E., Bonsignore, E. M., Walsh, G., & Atrash, Z. (2010). Toque: Designing a Cooking-Based Programming Language for And with Children. *Proceedings of the 28th International Conference on Human Factors in Computing Systems - CHI '10* (pp. 2417–2426). Atlanta, USA: ACM

Wegner, P. (1990). Concepts and paradigms of object-oriented programming. *ACM SIGPLAN OOPS Messenger, 1*(1), 7–87.

Weintrop, D., Hansen, A. K., Harlow, D. B., & Franklin, D. (2018). Starting from Scratch: Outcomes of Early Computer Science Learning Experiences and Implications for What Comes Next. *Proceedings of the 2018 ACM Conference on International Computing Education Research - ICER '18* (pp 142-150). Espoo, Finland: ACM.

Wilkinson, B. & Allen, M (2005). *Parallel Programming: Techniques and Applications Using Networked Workstations and Parallel Computers.* 2nd Edition, Pearson Prentice Hall.

Williams, L. & Upchurch, R.L. (2001), In Support of Student Pair-Programming. *Proceedings of the ACM SIGCSE '01 Conference* (pp. 327-331). Charlotte, USA: ACM.

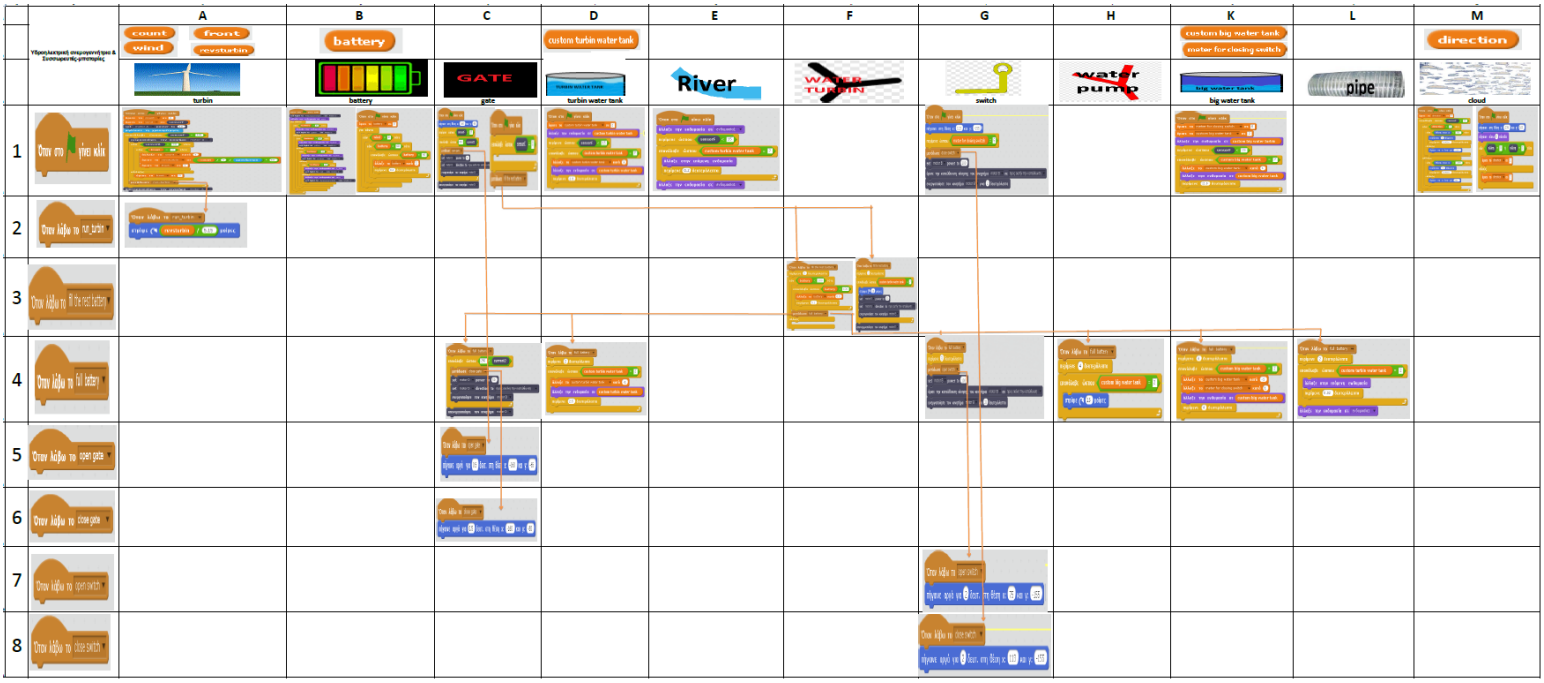
Wing, J. M. (2006). Computational thinking. *Communications of the ACM, 49*(3), 33-35.

Wing, J. M. (2008). Computational thinking and thinking about computing. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, 366*(1881), 3717–3725.

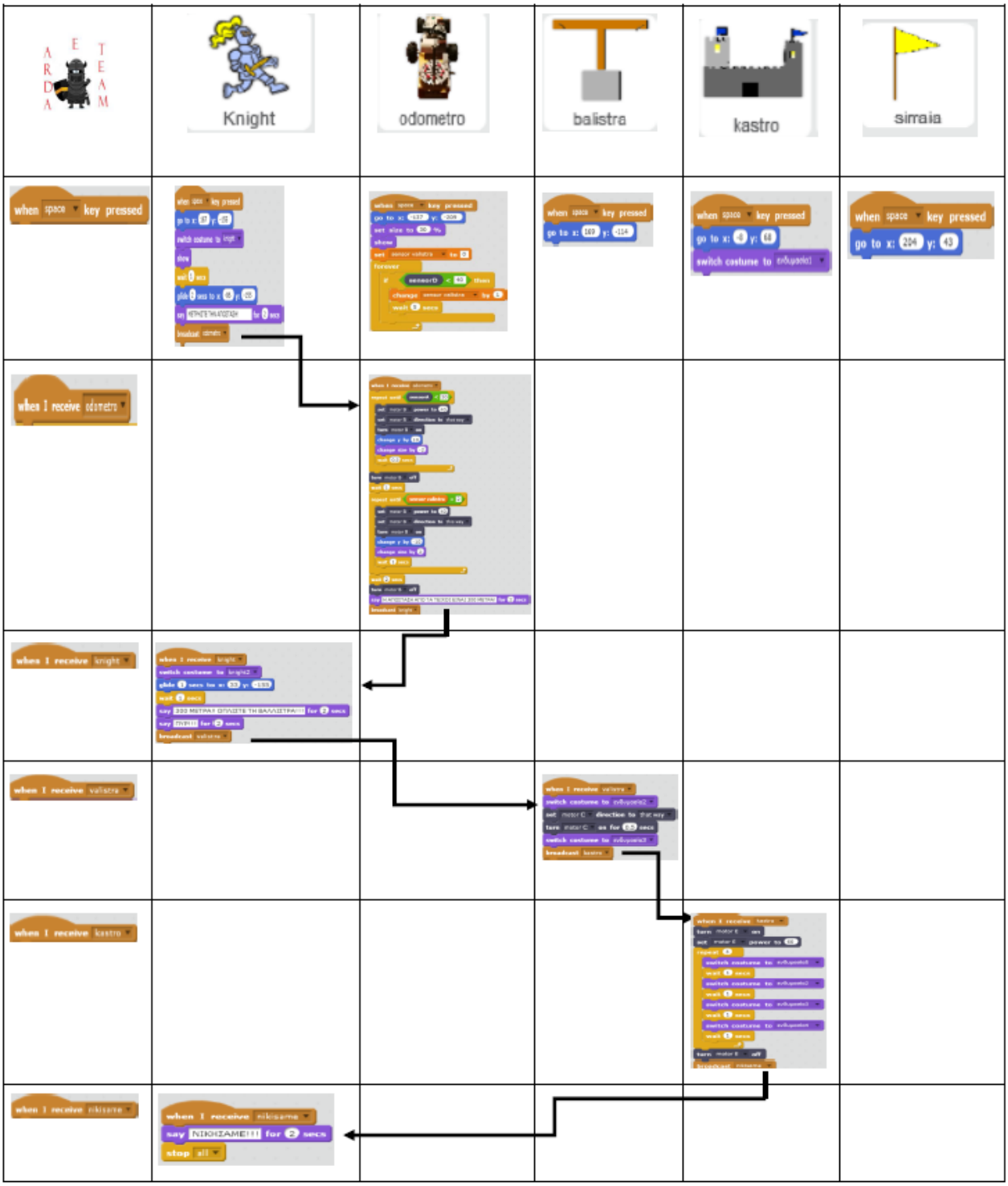
Zhang, J. X., Liu, L., Ordóñez de Pablos, P., & She, J. (2014). The auxiliary role of information technology in teaching: Enhancing programming course using alice. *International Journal of Engineering Education, 30*(3), 560–565.

ΠΑΡΑΡΤΗΜΑ

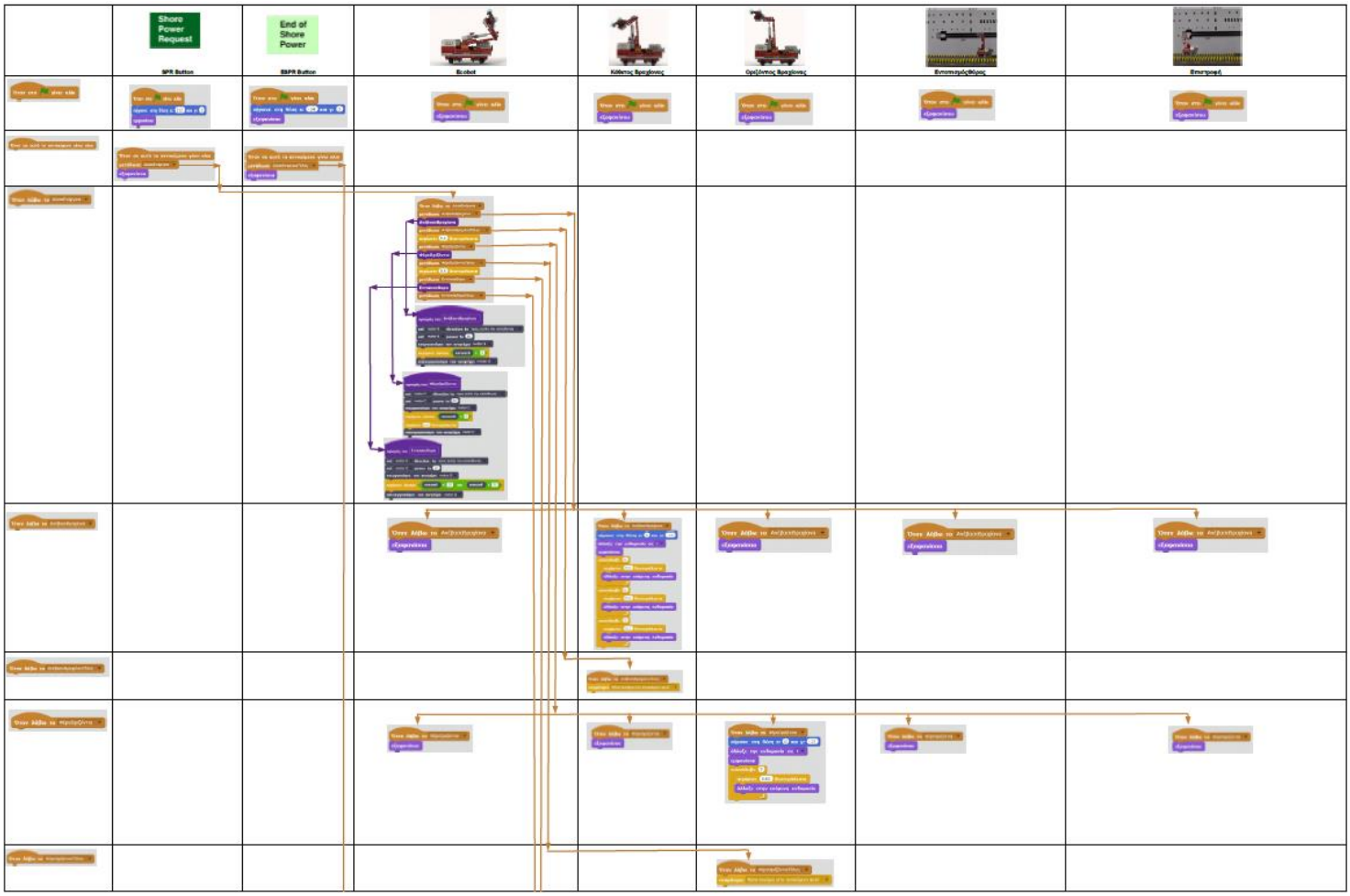
ΚωδικΌραμα 1



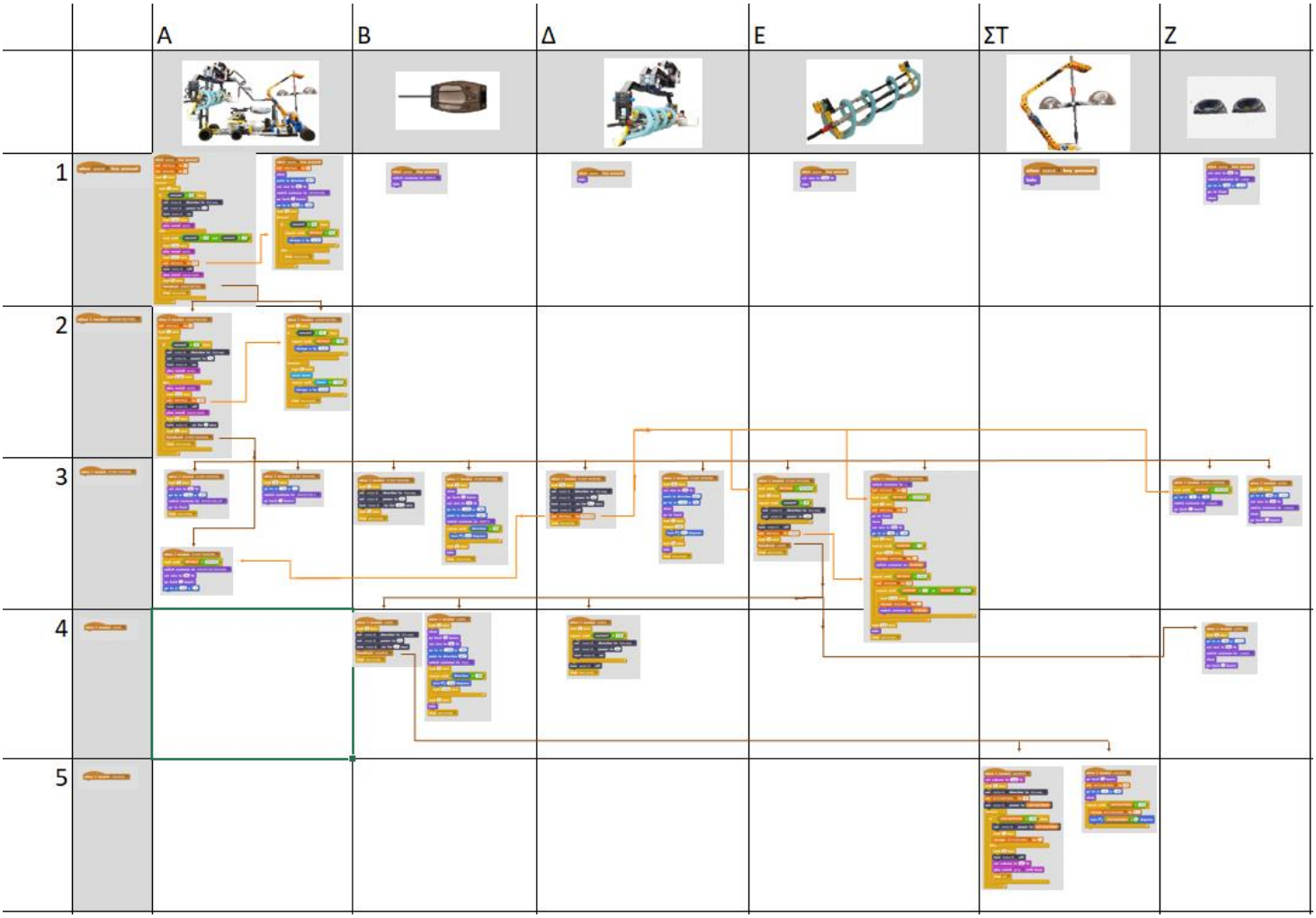
ΚωδικΌραμα 2



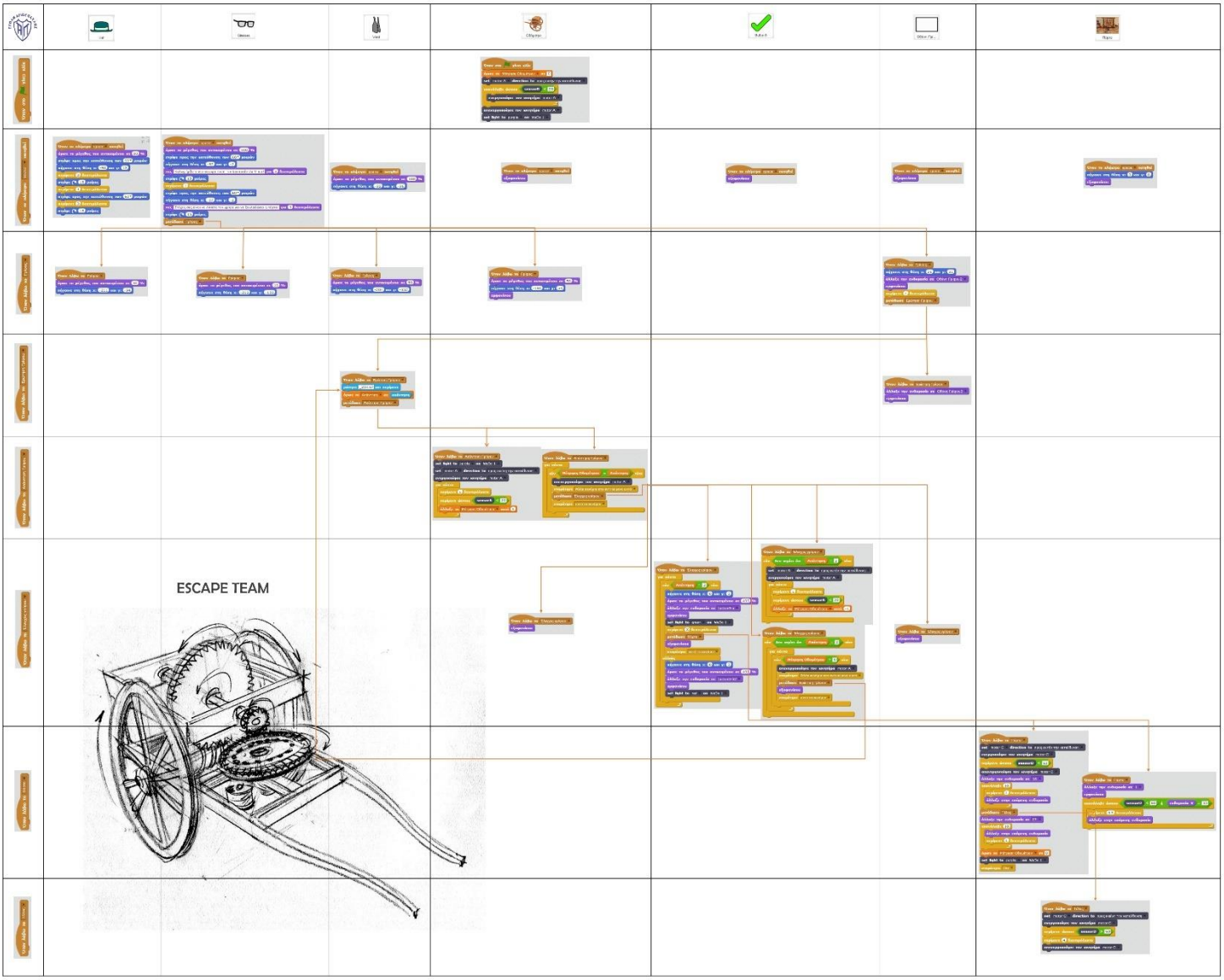
ΚωδικΌραμα 3



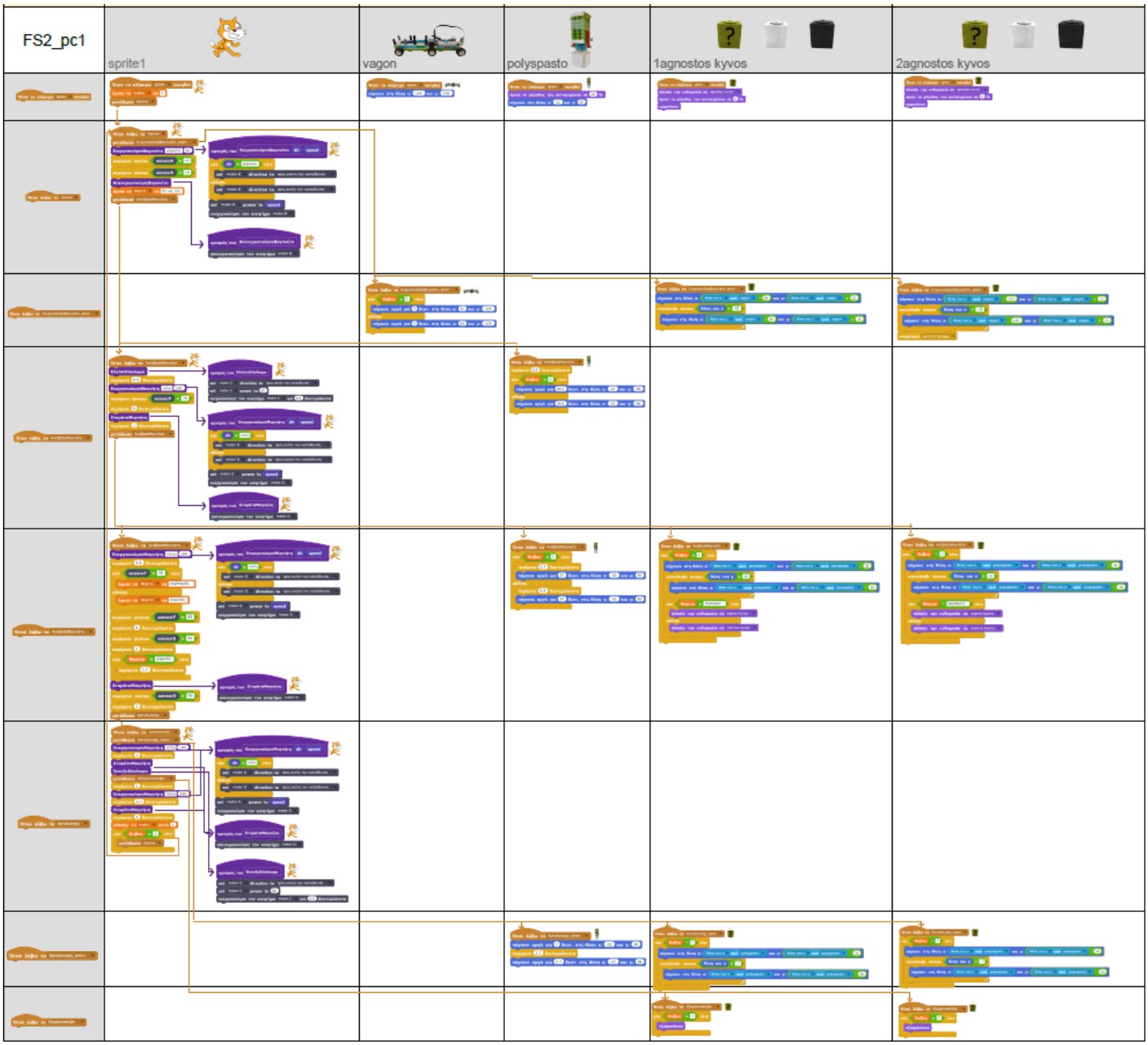
ΚωδικΌραμα 4



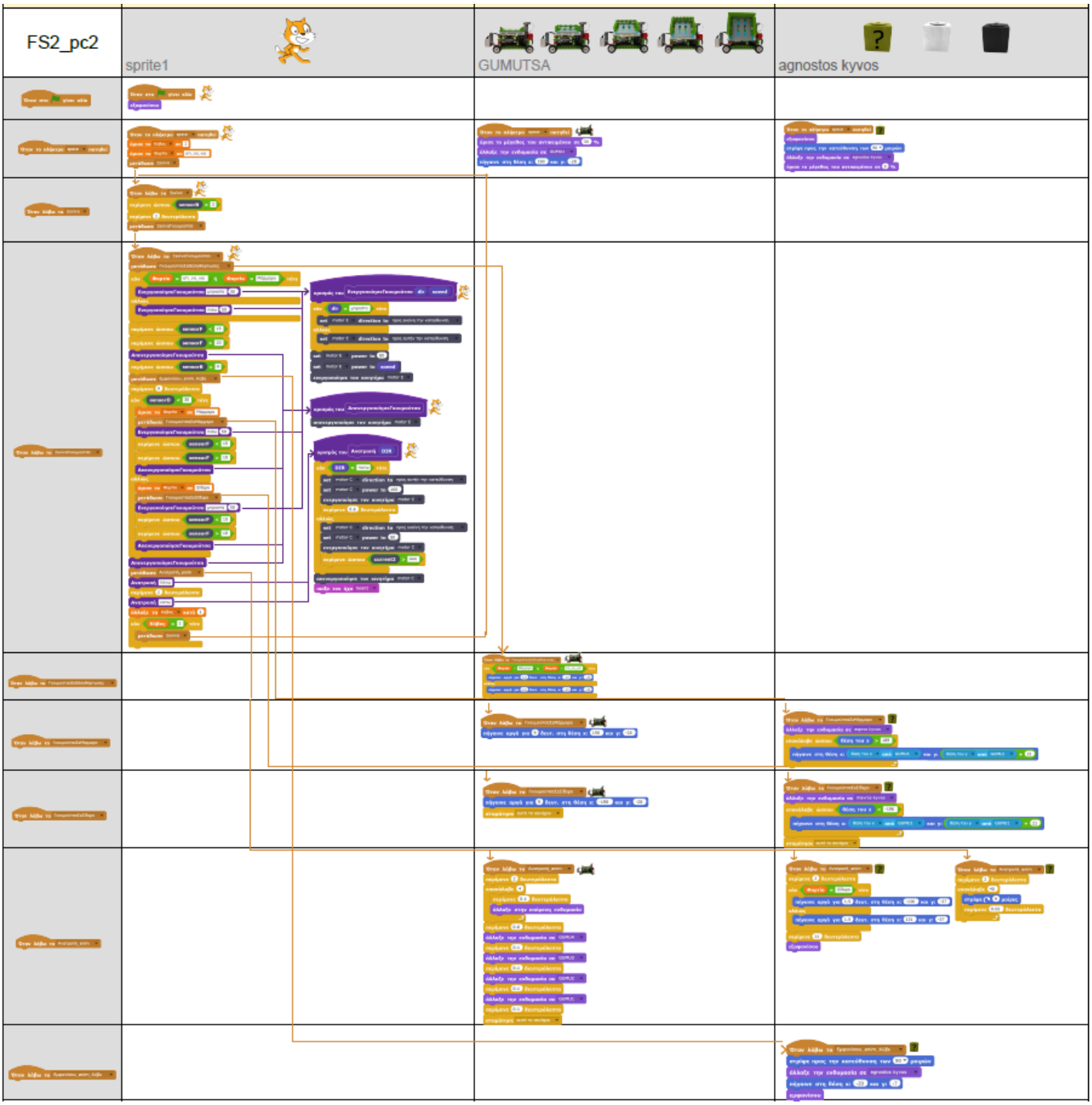
ΚωδικΌραμα 5



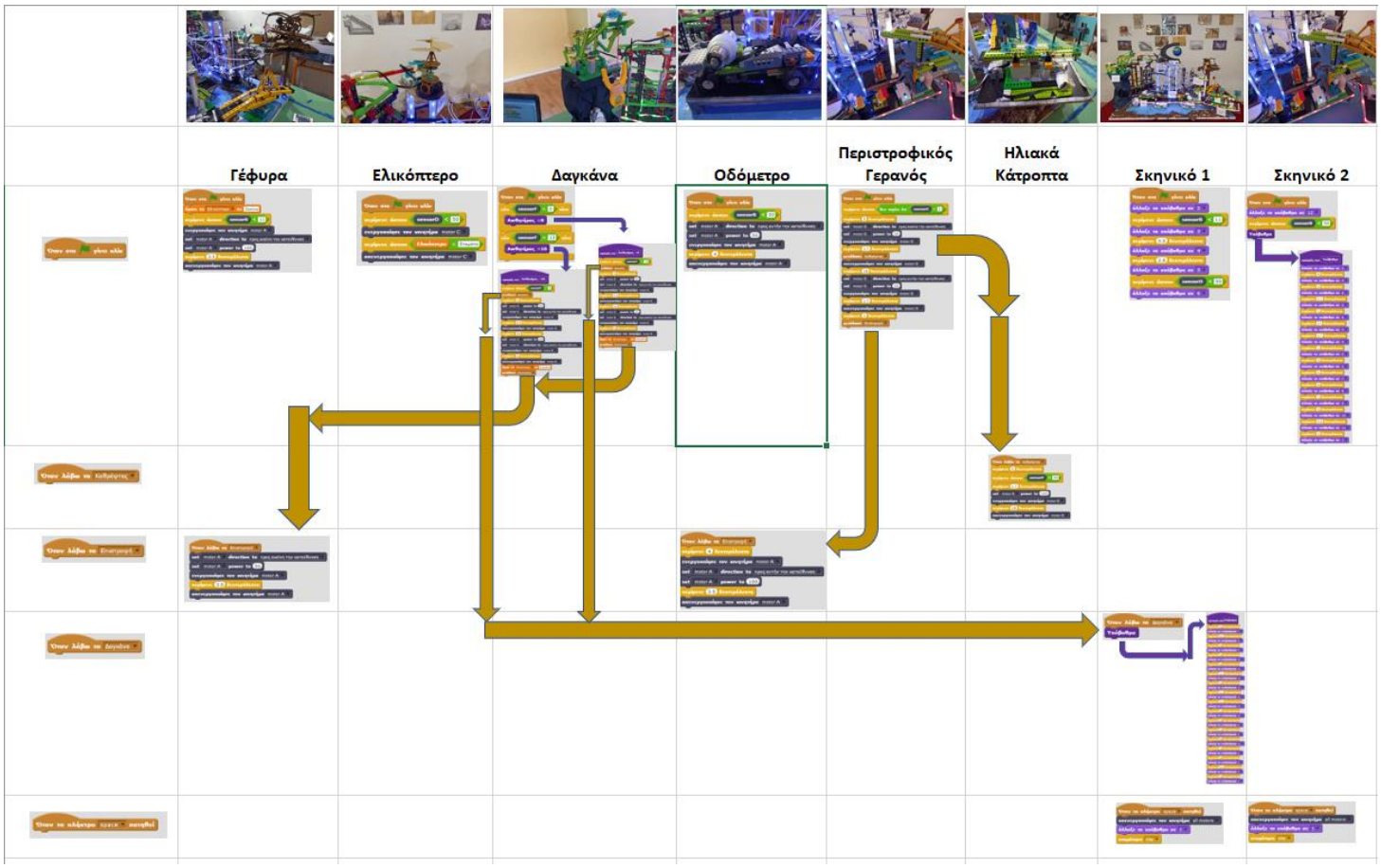
ΚωδικΌραμα 6



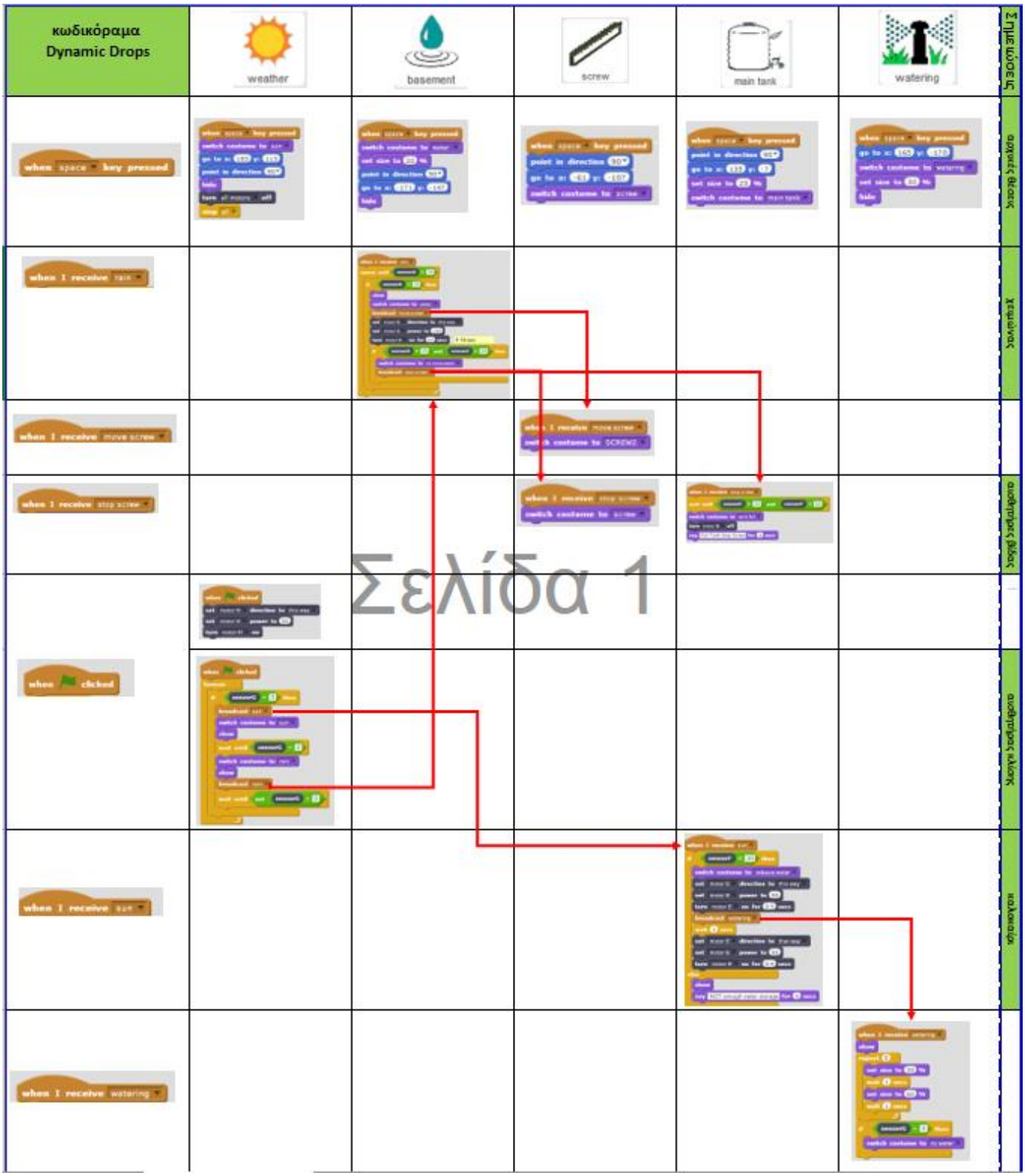
ΚωδικΌραμα 7



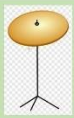
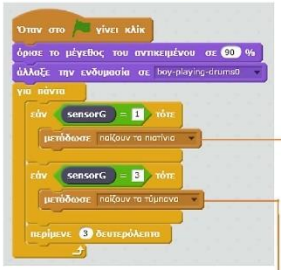
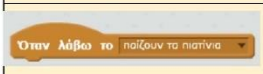
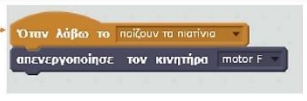
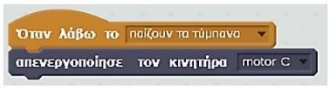
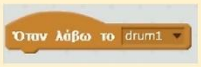
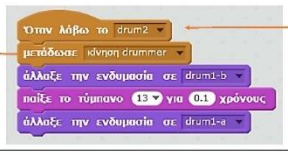
ΚωδικΌραμα 8



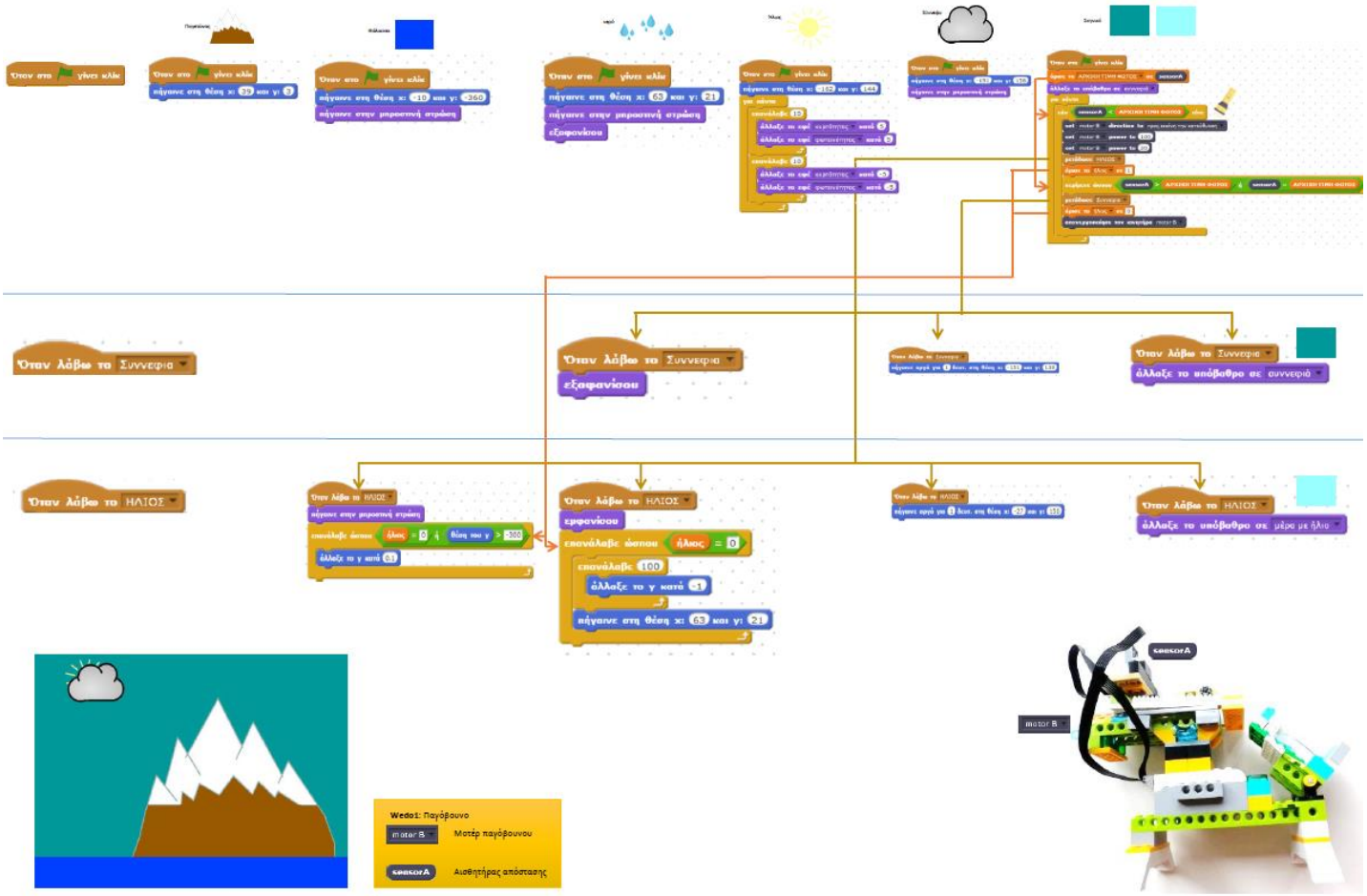
ΚωδικΌραμα 9



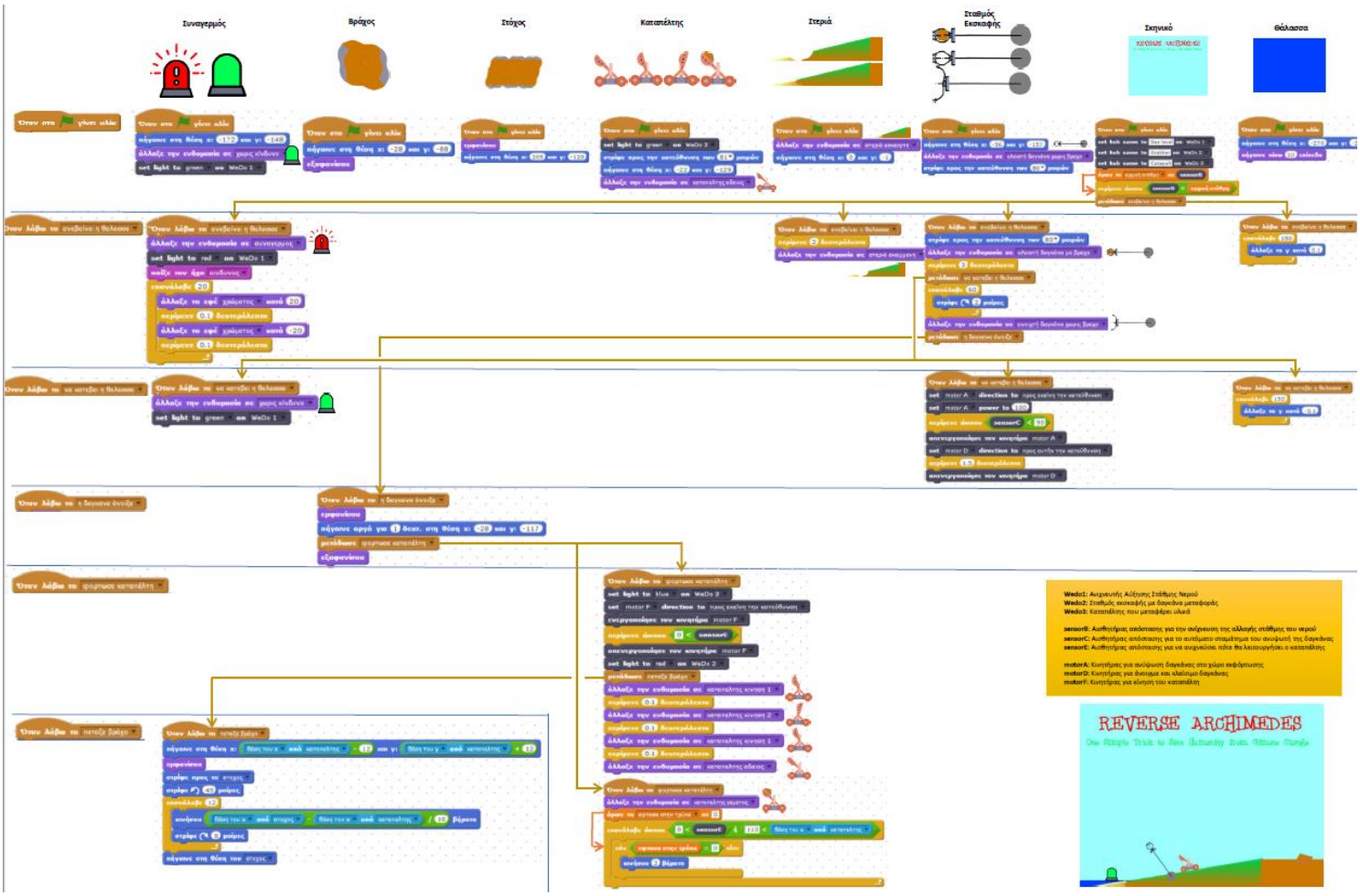
ΟΜΑΔΑ MUSIC-ROBOTS ΠΕΥΚΗΣ - ΚΩΔΙΚΟΡΑΜΑ ΓΙΑ ΤΟ ΠΡΟΓΡΑΜΜΑ ΤΟΥ WEDO 1 - Σεπτέμβρης 2020

ΟΜΑΔΑ MusicRobots ΜΟΥΣΙΚΑ ΡΟΜΠΟΤΣ DaVinci	 Τυμπανιστής	 Πιατίνια	 Τύμπανα
			
			
			
			
			
			

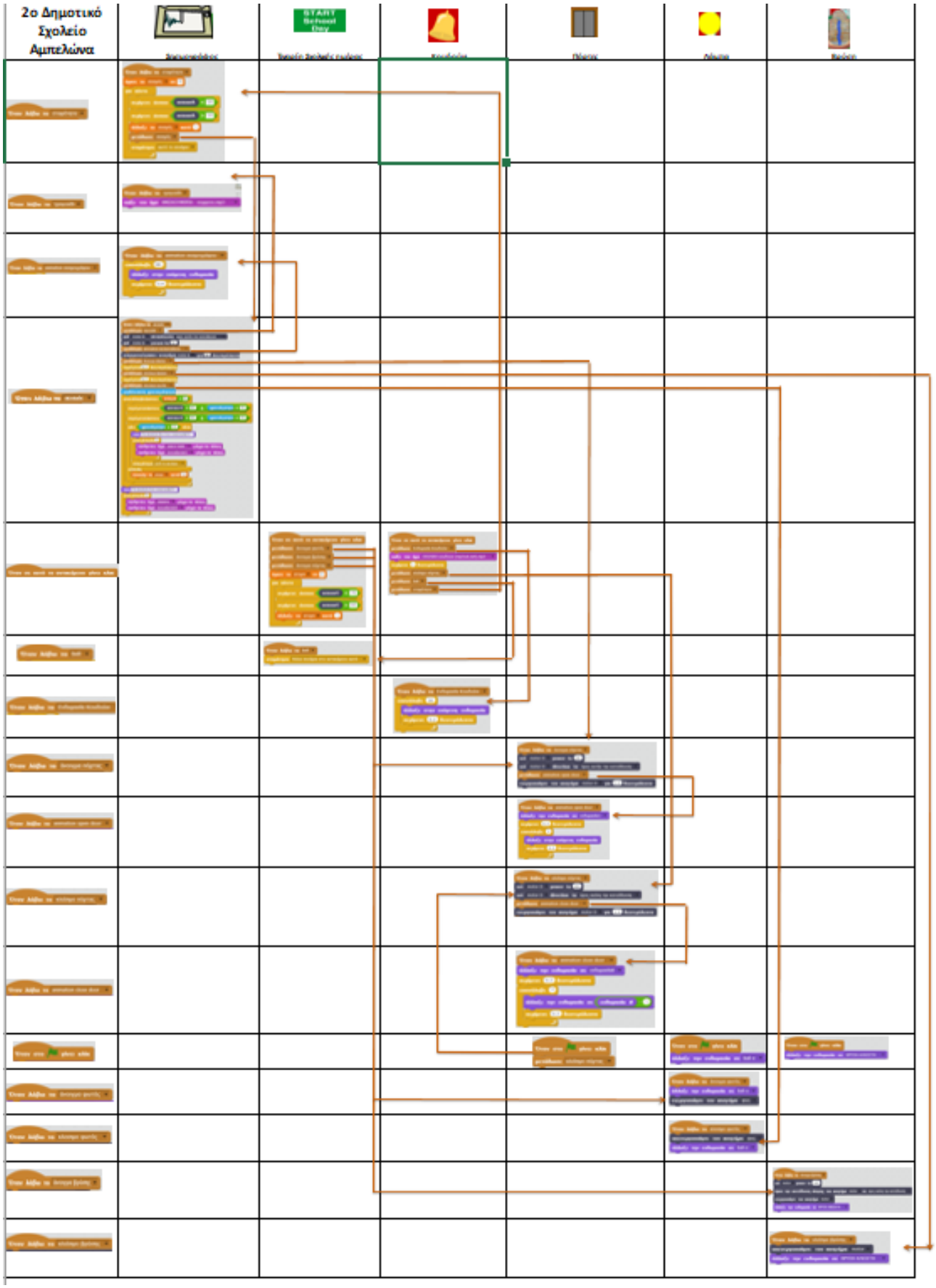
ΚωδικΌραμα 11



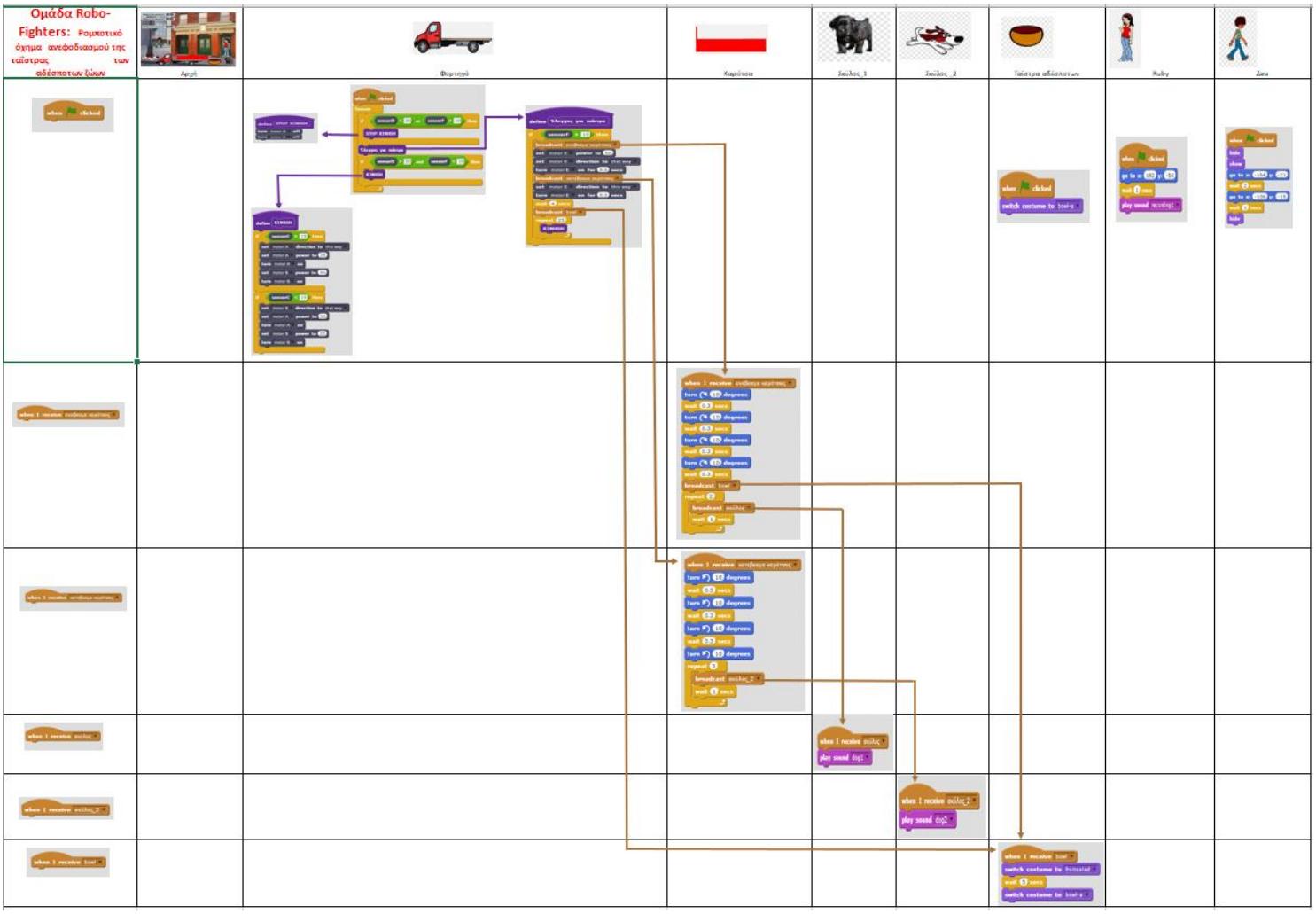
Κωδικόραμα 12



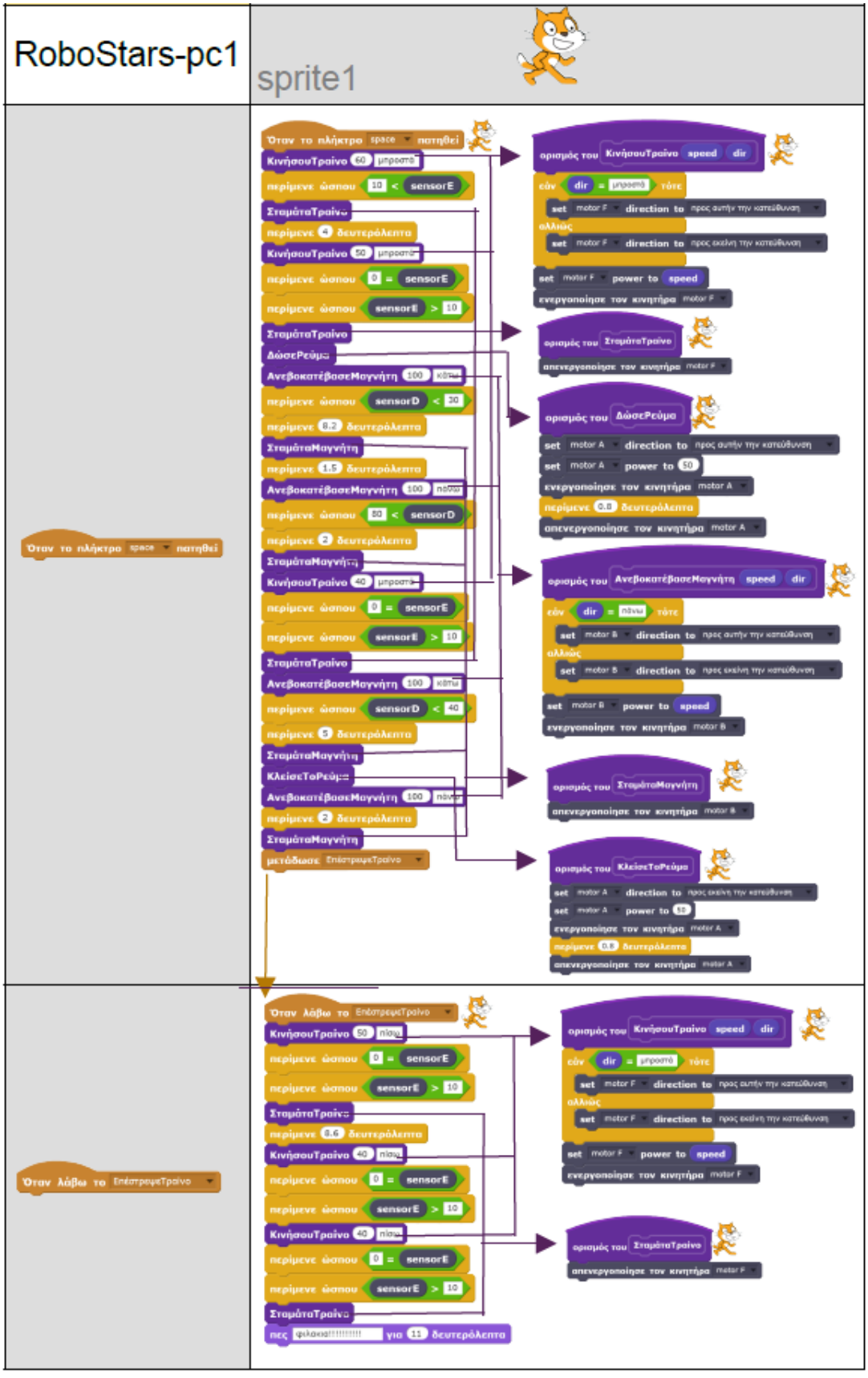
ΚωδικΌραμα 13



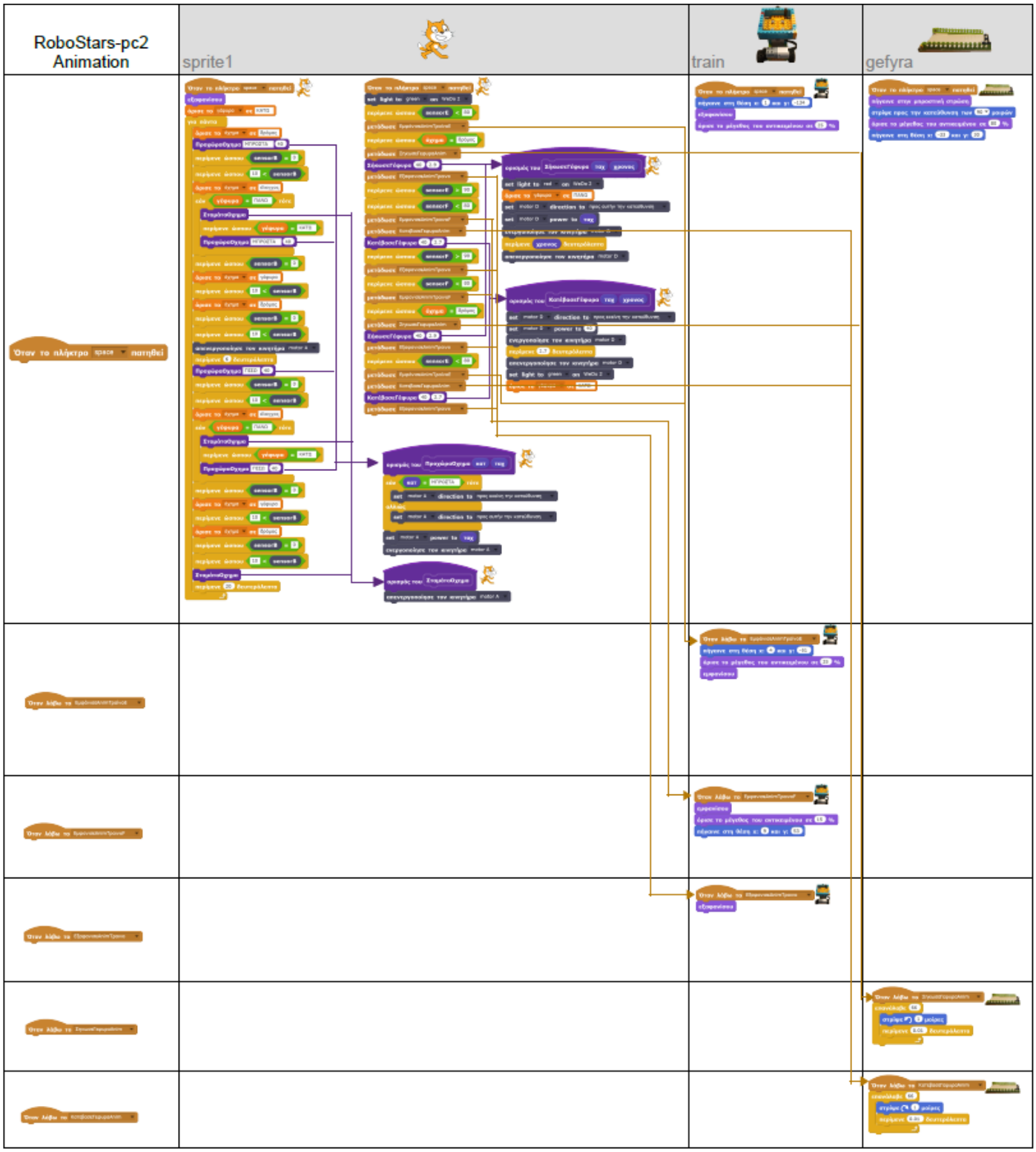
Κωδικόραμα 14



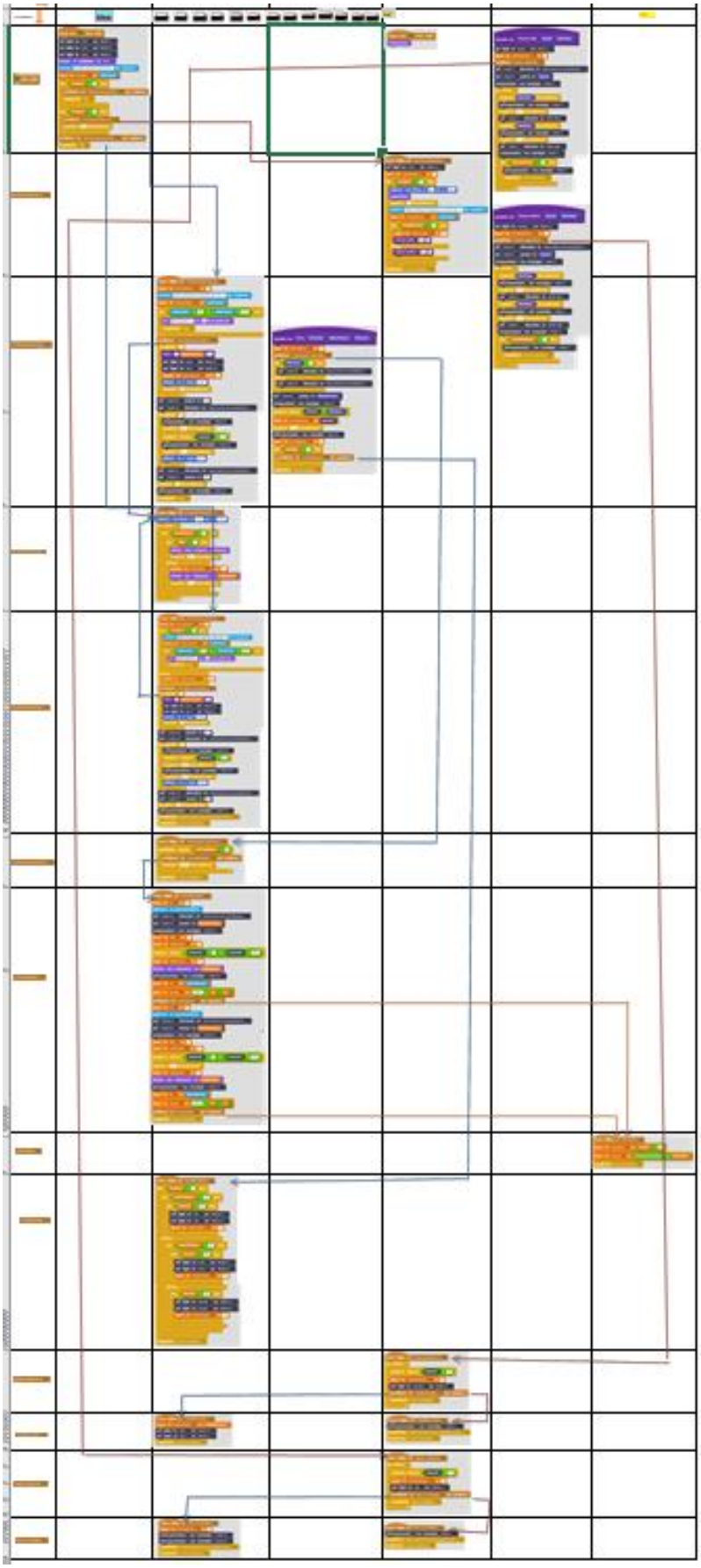
Κωδικόραμα 15



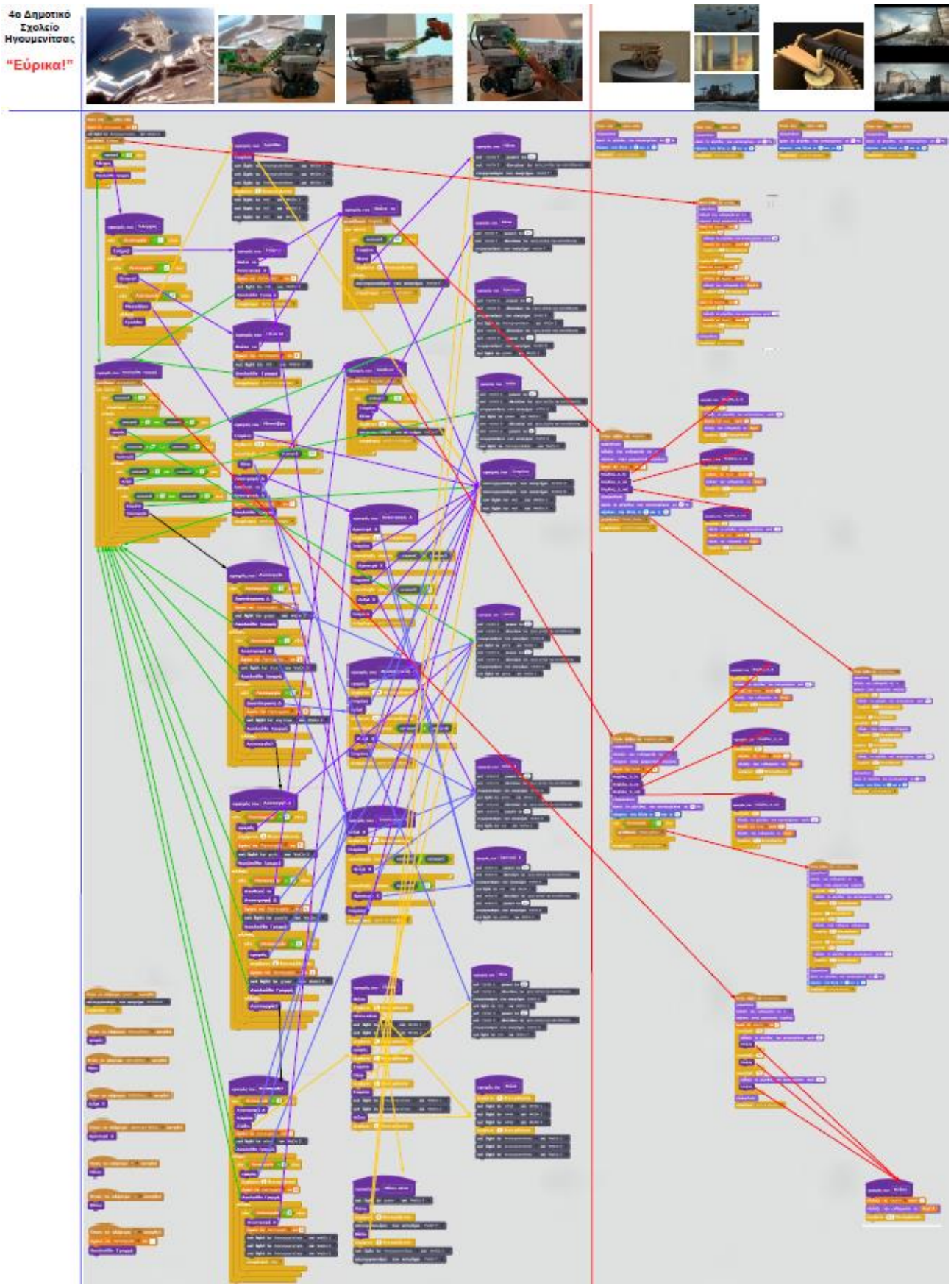
Κωδικόραμα 16



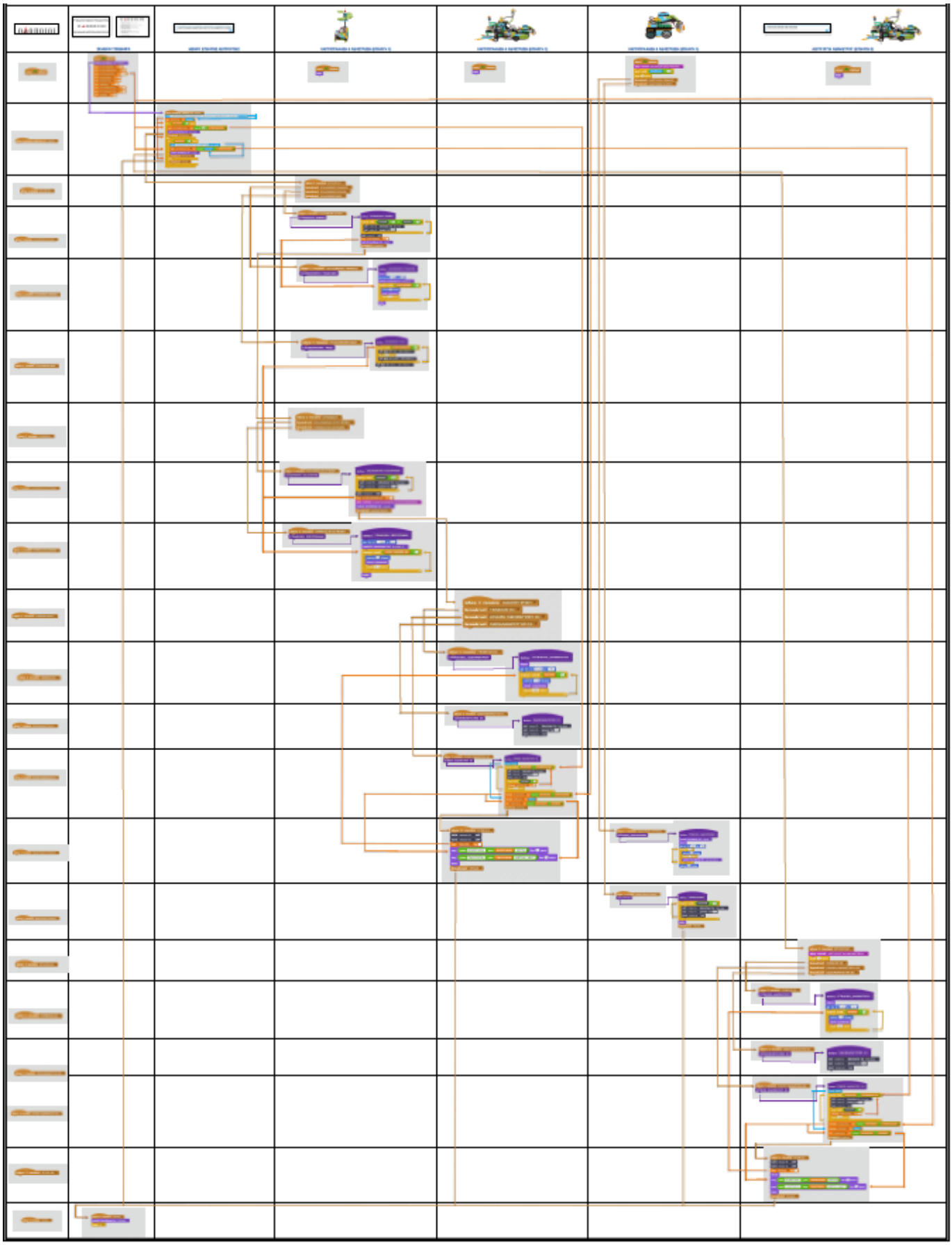
ΚωδικΌραμα 17



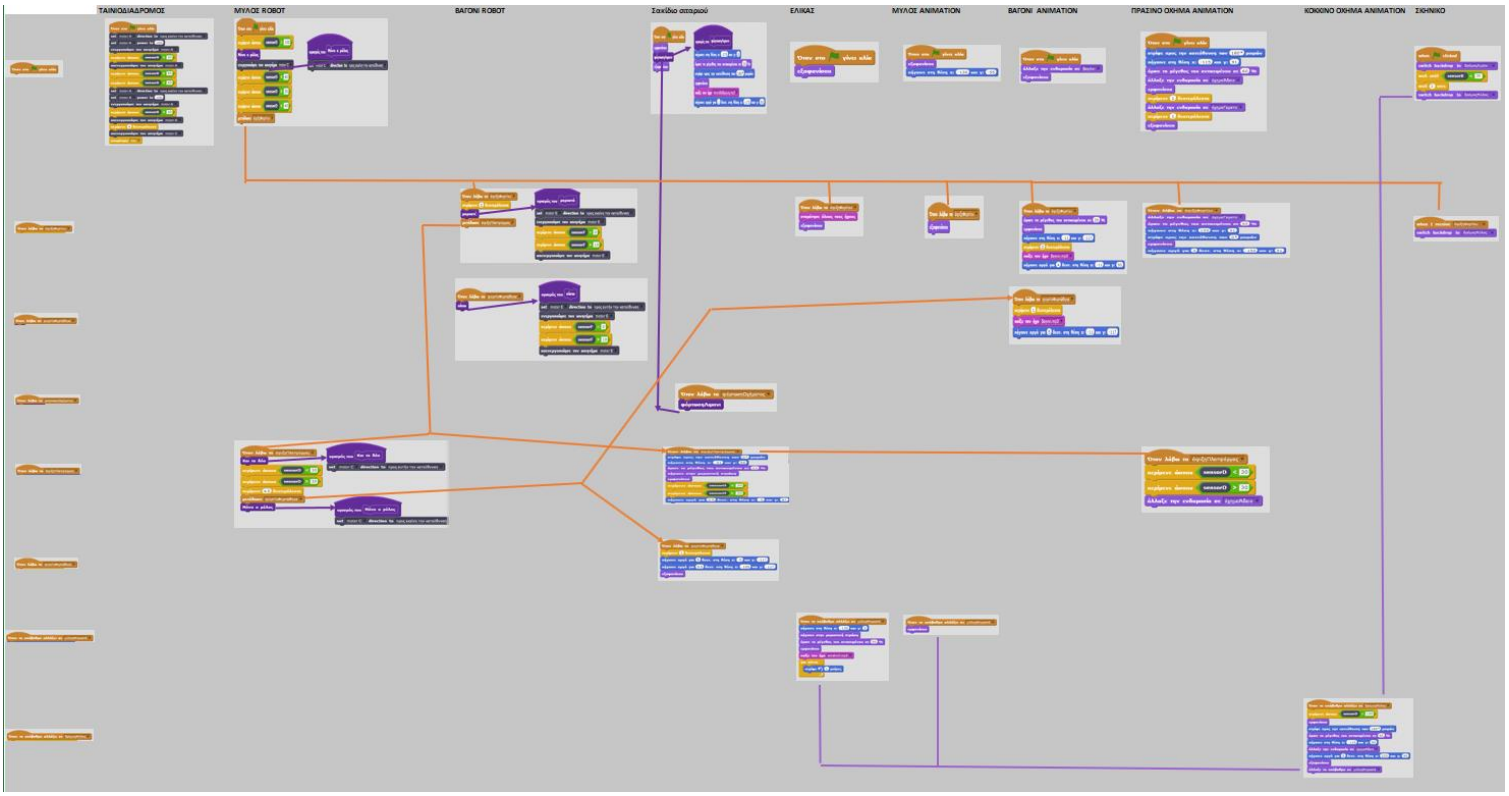
ΚωδικΌραμα 18



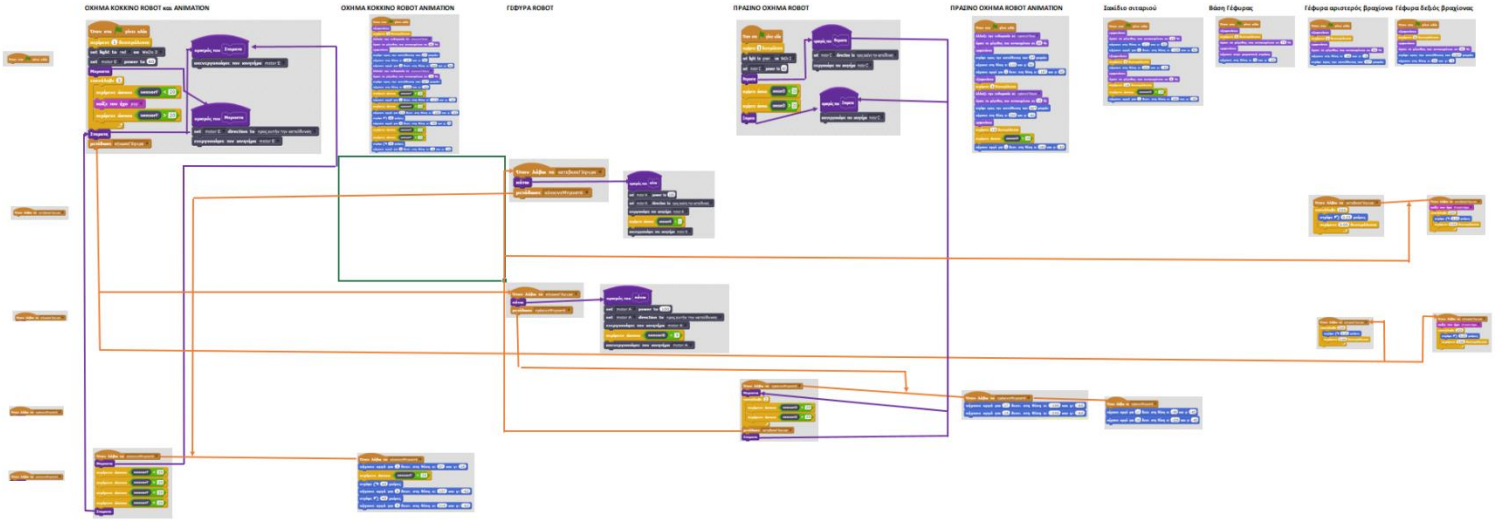
ΚωδικΌραμα 19



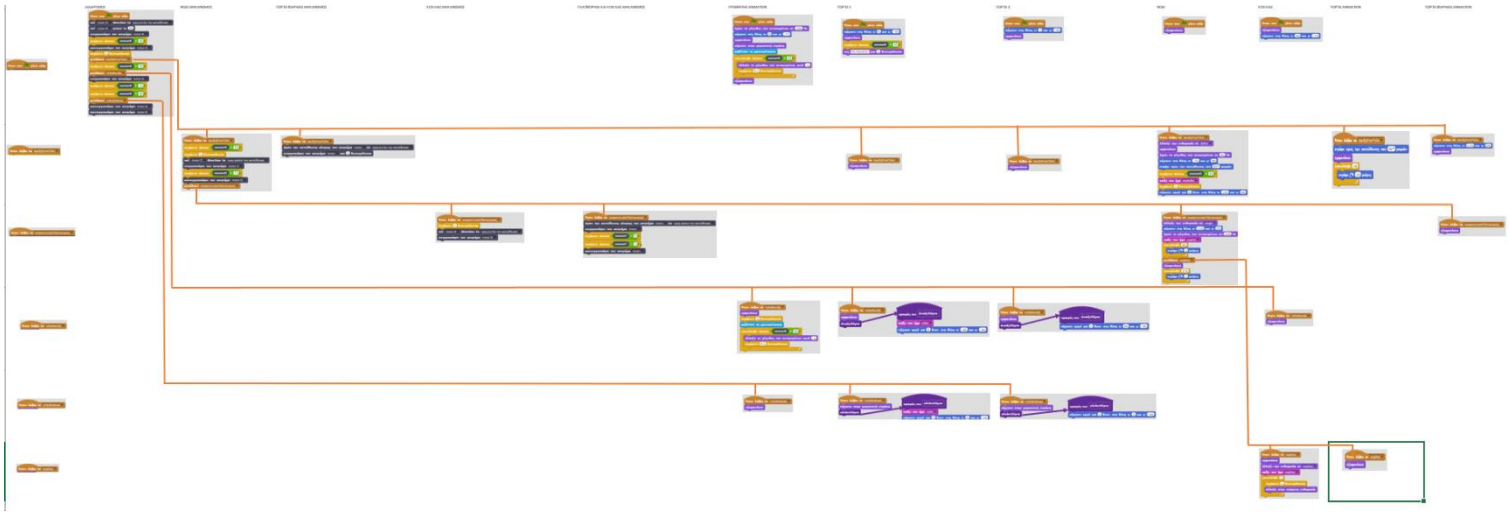
ΚωδικΌραμα 20



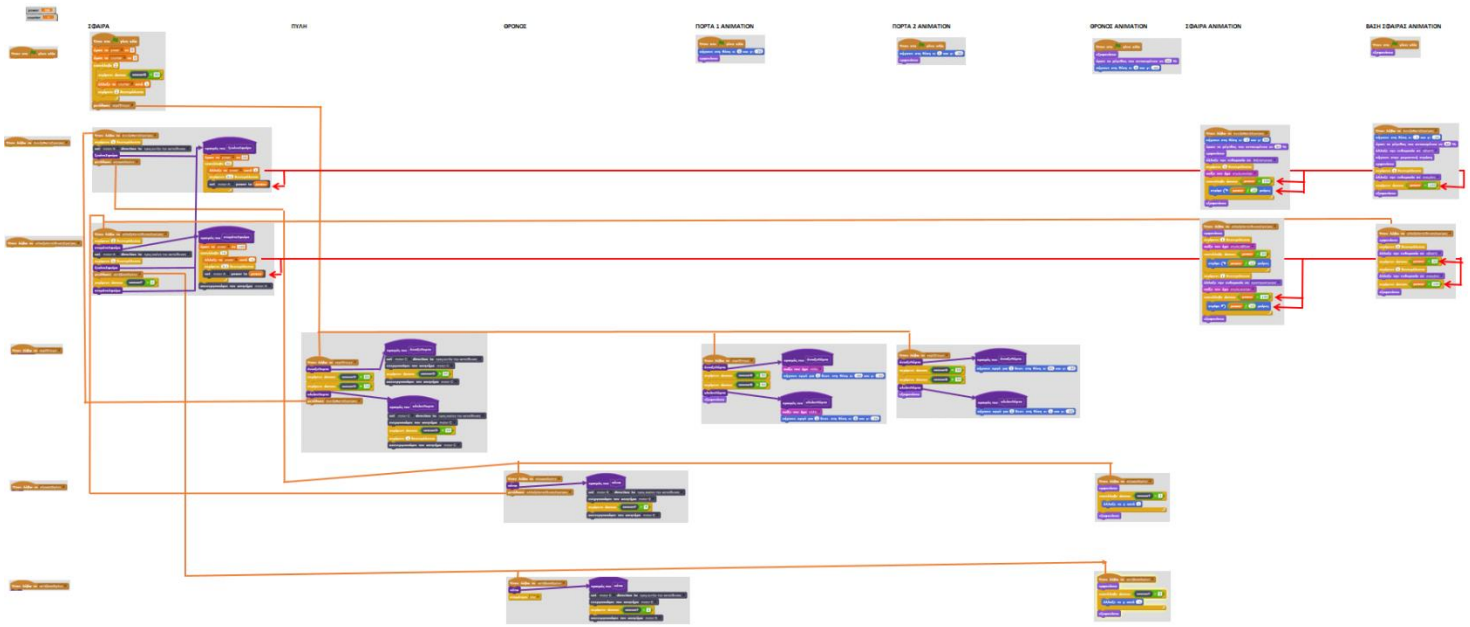
ΚωδικΌραμα 21



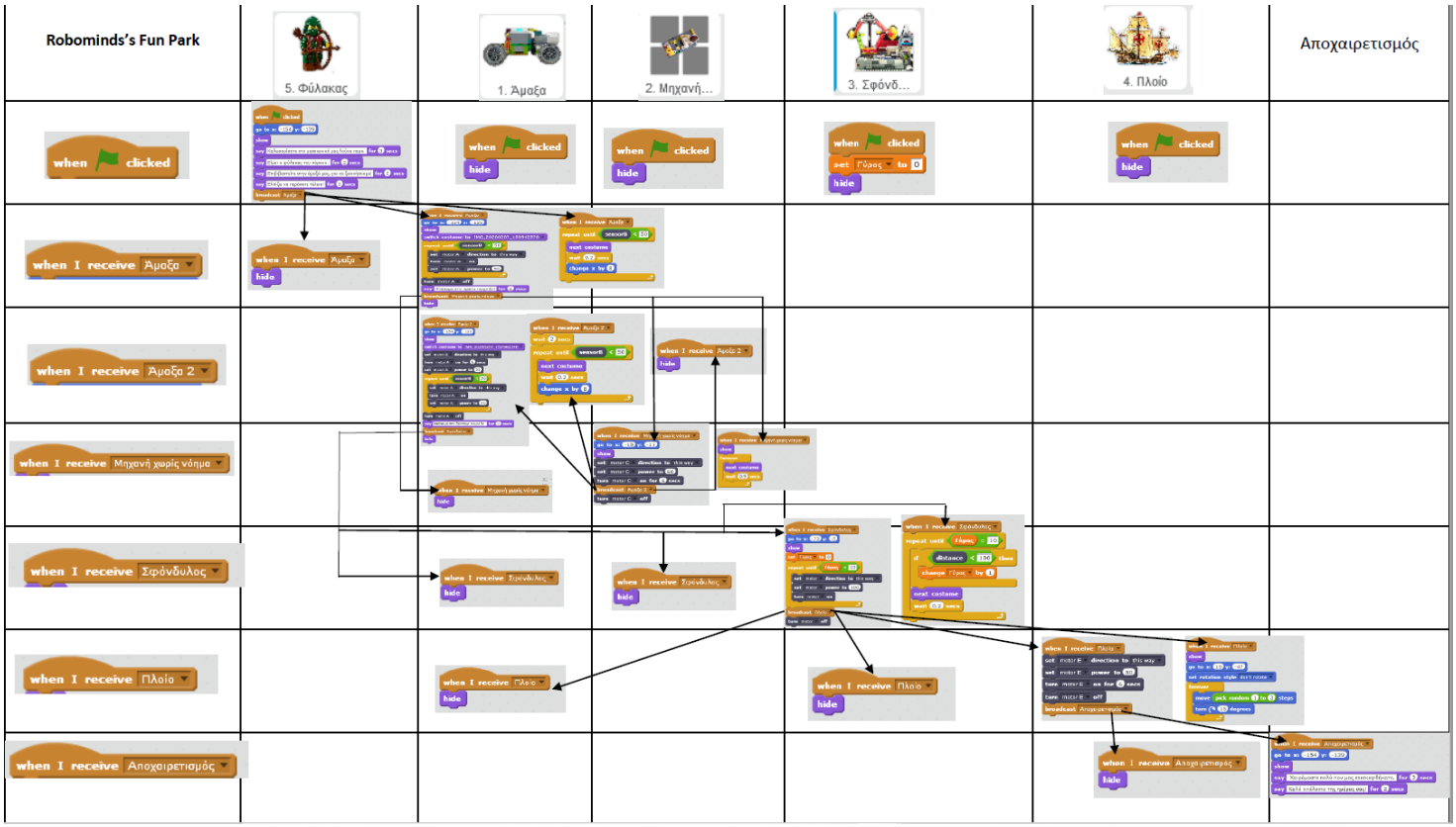
ΚωδικΌραμα 22



ΚωδικΌραμα 23



ΚωδικΌραμα 24



Κωδικόραμα 25