ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΣΥΣΤΗΜΑΤΩΝ ΜΕΤΑΔΟΣΗΣ ΠΛΗΡΟΦΟΡΙΑΣ
ΚΑΙ ΤΕΧΝΟΛΟΓΙΑΣ ΥΛΙΚΩΝ

Ανάπτυξη Πειραματικού Πρωτοκόλλου για Ανάλυση Χωρικής Αναπαράστασης & Προσανατολισμού υπό Συνθήκη Παρεμπόδισης Όρασης με χρήση Ηλεκτροεγκεφαλογραφήματος

Development of an Experimental Protocol for Task-Based EEG Analysis on Spatial Representation & Orientation under Vision Deprivation

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Δανάη Λιάκου

Επιβλέπων: Γεώργιος Ματσόπουλος
Καθηγητής Ε.Μ.Π.

Αθήνα, Οκτώβριος 2025

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΣΥΣΤΗΜΑΤΩΝ ΜΕΤΑΔΟΣΗΣ ΠΛΗΡΟΦΟΡΙΑΣ &
ΤΕΧΝΟΛΟΓΙΑΣ ΥΛΙΚΩΝ

Ανάπτυξη Πειραματικού Πρωτοκόλλου για Ανάλυση Χωρικής Αναπαράστασης & Προσανατολισμού υπό Συνθήκη Παρεμπόδισης Όρασης με χρήση Ηλεκτροεγκεφαλογραφήματος

Development of an Experimental Protocol for Task-Based EEG Analysis on Spatial Representation & Orientation under Vision Deprivation

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Δανάη Λιάκου

Επιβλέπων: Γεώργιος Ματσόπουλος
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 16η Οκτωβρίου 2025.

.....	.....	.....
Γεώργιος Ματσόπουλος	Παναγιώτης Τσανάκας	Αθανάσιος Δ. Παναγόπουλος
Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.

Αθήνα, Οκτώβριος 2025

.....

Δανάη Λιάκου

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Περίληψη

Στην παρούσα διπλωματική εργασία αναπτύσσεται και ελέγχεται η εφαρμογή ενός πειράματος, το οποίο μέσα από τη χρήση επιφανειακού ηλεκτροεγκεφαλογραφήματος μελετά τη χωρική αναπαράσταση και τον χωρικό προσανατολισμό υπό τη συνθήκη της παρεμπόδισης της όρασης.

Αρχικά, μετά την εισαγωγή, αναλύεται το θεωρητικό υπόβαθρο της εγκεφαλικής λειτουργίας, παρατίθενται οι ορισμοί της χωρικής αναπαράστασης και του χωρικού προσανατολισμού, ενώ επίσης επεξηγείται η ηλεκτροεγκεφαλογραφία από τη δημιουργία του σήματος μέχρι την επεξεργασία του.

Στη συνέχεια, γίνεται αναφορά στο πειραματικό πρωτόκολλο και αναλύεται ο σκοπός του πειράματος και η ακριβής διαδικασία που ακολουθήθηκε για τη λήψη των απαιτούμενων σημάτων, την επεξεργασία και διερμηνεία τους.

Τέλος, παρουσιάζονται τα αποτελέσματα και τα σχετικά συμπεράσματα, μαζί με την αντιπαραβολή τους με άλλες παρόμοιες μελέτες πάνω στο ίδιο αντικείμενο.

Λέξεις Κλειδιά

ηλεκτροεγκεφαλογράφημα, χωρική μνήμη, χωρική αναπαράσταση, όραση, χωρικός προσανατολισμός, απουσία όρασης, παρεμπόδιση όρασης, ανάλυση σήματος, ρυθμοί ΗΕΓ

Abstract

In this diploma thesis, an experiment is developed and tested to study spatial representation and spatial orientation under the condition of visual deprivation through the use of electroencephalography (EEG).

First, after the introduction, the related theoretical background is presented, including the brain function, the definitions of spatial representation and spatial orientation, as well as the generation, recording and processing of the EEG signal.

Afterwards, the experimental protocol and the purpose of the experiment are presented, followed by the exact procedure implemented for signal acquisition, processing, and interpretation.

Finally, results and conclusions are presented, together with their potential association with other similar studies.

Key Words

electroencephalography, spatial memory, spatial representation, spatial orientation, vision, visual deprivation, obstruction of vision, signal processing, EEG rhythms

Περιεχόμενα

Περιεχόμενα 7
1. Αρκτικόλεξα 9
2. Ευρετήριο Εικόνων 10
3. Ευρετήριο Πινάκων 11
4. Εισαγωγή 12
5. Εγκεφαλική Δραστηριότητα – Θεωρητικό Υπόβαθρο 13
   5.1. Λειτουργικός Διαχωρισμός Εγκεφάλου 13
   5.2. Χωρική Αναπαράσταση & Χωρικός Προσανατολισμός 15
   5.3. Συνδρομή Όρασης & Λειτουργία υπό Απουσία Οπτικών Ερεθισμάτων 18
6. Ηλεκτροεγκεφαλογραφία (EEG) 23
   6.1. Εισαγωγή 23
   6.2. Φυσιολογία Δημιουργίας Σήματος 24
   6.3. Βασικοί Ρυθμοί Εγκεφαλογραφήματος 29
      6.3.1. Ρυθμός α (alpha rhythm) 30
      6.3.2. Ρυθμός β (beta rhythm) 30
      6.3.3. Ρυθμός γ (gamma rhythm) 31
      6.3.4. Ρυθμός δ (delta rhythm) 31
      6.3.5. Ρυθμός θ (theta rhythm) 31
   6.4. Λήψη Σήματος 32
      6.4.1. Διαδικασία 32
      6.4.2. Εξοπλισμός 34
   6.5. Επεξεργασία Σήματος 36
      6.5.1. Προεπεξεργασία Δεδομένων 36
      6.5.2. Ανάλυση Δεδομένων 39
7. Πειραματικό Πρωτόκολλο 41
   7.1. Σκοπός Πειράματος 41
   7.2. Προηγούμενα Ευρήματα 41
   7.3. Περιγραφή Πειράματος 43
   7.4. Λήψη Δεδομένων 55
8. Ανάλυση Δεδομένων 56
9. Αποτελέσματα 60
10. Συμπεράσματα 61
11. Βιβλιογραφία 63
12. Παράρτημα Α: Κώδικας PsychoPy 69
   Δομή Πειράματος 70
   Ροή Πειράματος 86

Τερματισμός Πειράματος 129
13. Παράρτημα Β: Κώδικας MATLAB 130
Αρχικοποίηση 130
Φόρτωση και Μετασχηματισμός Δεδομένων 130
Εξαγωγή Μετρήσεων και Συμπερασμάτων 132

1. Αρκτικόλεξα

ΗΕΓ	Ηλεκτροεγκεφαλογράφημα
BCI	Brain-Computer Interface
ECoG	Electrocorticogram
EDF	Electrochemical Driving Force
EEG	Electroencephalogram
EPSP	Excitatory PostSynaptic Potential
GUI	Graphic User Interface
ICA	Independent Component Analysis
iEEG	Intracranial Electroencephalogram
IPSP	Inhibitory PostSynaptic Potential
MEG	Magnetoencephalogram

2. Ευρετήριο Εικόνων

Εικόνα 1 Διάγραμμα του ανθρώπινου εγκεφάλου, που δείχνει την παρεγκεφαλίδα (cerebellum), το εγκεφαλικό στέλεχος (brainstem) και τον <τελικό> εγκέφαλο (cerebrum) 13
Εικόνα 2 Απεικόνιση των κύριων λοβών του ανθρώπινου εγκεφάλου 14
Εικόνα 3 Χάρτης των περιοχών του εγκεφάλου σύμφωνα με τον χάρτη του Brodmann 15
Εικόνα 4 Απεικόνιση της Εργαζόμενης Μνήμης 15
Εικόνα 5 Απεικόνιση του ιππόκαμπου 16
Εικόνα 6 Σχεδιάγραμμα διαφορών μεταξύ εγωκεντρικής και αλλοκεντρικής αναπαράστασης 18
Εικόνα 7 Διακριτές ζώνες χωρικού περιβάλλοντος 20
Εικόνα 8 Σχέδιο ατόμου κατά την διάρκεια λήψης επιφανειακού ηλεκτροεγκεφαλογραφήματος 23
Εικόνα 9 Απεικόνιση λήψης ενδοκρανιακού εγκεφαλογραφήματος 23
Εικόνα 10 Δυναμικό δράσης 27
Εικόνα 11 Απεικόνιση της περιόδου ανερεθιστότητας 28
Εικόνα 12 Ρυθμός α 30
Εικόνα 13 Ρυθμός β 30
Εικόνα 14 Ρυθμός γ 31
Εικόνα 15 Ρυθμός δ 31
Εικόνα 16 Ρυθμός θ 31
Εικόνα 17 Σύστημα καταγραφής ηλεκτοεγκεφαλογραφήματος 32
Εικόνα 18 Σύστημα 10/20 35
Εικόνα 19 Σύστημα 10/10 36
Εικόνα 20 Διαχωρισμός σήματος ΗΕΓ σε σήματα αναλόγως με την πηγή τους 37
Εικόνα 21 Επεξήγηση της ανάλυσης ανεξάρτητων συνστωσών (ICA) 38
Εικόνα 22 Σχηματική αναπαράσταση του πειραματικού πρωτοκόλλου 43
Εικόνα 23 Σχηματική αναπαράσταση των φάσεων του πειράματος. Με κίτρινο οι οδηγίες της κάθε φάσης και με κόκκινο η πειραματική διαδικασία 44
Εικόνα 24 Δείγμα του τρόπου προβολής των οδηγιών. Ίδια διαδικασία πριν από κάθε φάση απλώς με διαφορετικό κείμενο 45
Εικόνα 25 Φάση 1: Ο ήχος της έναρξης , η προβολή της εικόνας του κεντραρισμένου μαύρου σταυρού και ο ήχος της λήξης 46
Εικόνα 26 Φάση 2: Ο ήχος της έναρξης, παύση 5 δευτερολέπτων και ο ήχος της λήξης 46
Εικόνα 27 Φάση 3: Ο ήχος της έναρξης, η εμφάνιση όλων των 24 κελιών του πλέγματος στην οθόνη για 30 δευτερόλεπτα και ο ήχος της λήξης 47
Εικόνα 28 Φάση 4: Η ανακοίνωση της θέσης που πρέπει να πιέσει ο συμμετέχοντας και η πίεση 48
Εικόνα 29 Φάση 4: Ο ήχος λήξης της επανάληψης και η πίεση του πλήκτρου 'space' από τον επιτηρητή 49
Εικόνα 30 Κάτοψη από τον σχεδιασμό του πλέγματος 50
Εικόνα 31 Στιγμιότυπο από το λογισμικό PrusaSlicer για την εκτύπωση του σκελετού 51
Εικόνα 32 Εκτύπωση του σκελετού 51
Εικόνα 33 Τα 2 διακριτά κομμάτια του σκελετού ηλεκτροεγκεφαλογράφου εκτυπωμένα 52
Εικόνα 34 Σχεδίαση αρχείου τρισδιάστατης εκτύπωσης με τα υπόλοιπα εξαρτήματα της κάσκας 52
Εικόνα 35 Εκτύπωση των εξαρτημάτων - Αρχικό στάδιο 53
Εικόνα 36 Ολοκληρωμένη κάσκα ηλεκτροεγκεφαλογράφου OpenBCI 53
Εικόνα 37 Σύνδεση ηλεκτροδίων με το σύστημα Cyton+Daisy 54

3. Ευρετήριο Πινάκων

Πίνακας 1 Χαρακτηριστικά προσανατολισμού ανάλογα με ικανότητα όρασης..... **Σφάλμα! Δεν έχει οριστεί σελιδοδείκτης.**
Πίνακας 2 Ηλεκτρόδια ηλεκτροεγκεφαλογράφου OpenBCI ..... **Σφάλμα! Δεν έχει οριστεί σελιδοδείκτης.**

4. Εισαγωγή

Σκοπός της εργασίας είναι να μελετήσει τη διαφοροποίηση στην εγκεφαλική δραστηριότητα κατά τον προσανατολισμό και την εκτέλεση εργασιών σε ένα «γνωστό» χώρο που κωδικοποιήθηκε με και χωρίς αξιοποίηση οπτικών ερεθισμάτων, καθώς και να διακρίνει τις διαφορές στη λειτουργία του εγκεφάλου κατά τη φάση της κωδικοποίησης με καθέναν από τους δύο προαναφερθέντες τρόπους. Για τον σκοπό αυτό, αξιοποιείται η τεχνική του επιφανειακού ηλεκτροεγκεφαλογραφήματος, που είναι ένας πολύ διαδεδομένος τρόπος παρακολούθησης και καταγραφής της ηλεκτρικής δραστηριότητας που λαμβάνει χώρα στο τριχωτό της κεφαλής και αντιπροσωπεύει μακροσκοπικά την εγκεφαλική δραστηριότητα.

Η χωρική αναπαράσταση και ο χωρικός προσανατολισμός είναι βασικοί μηχανισμοί που χρησιμοποιούνται από τον άνθρωπο σε καθημερινή βάση για την αφομοίωση και καλύτερη εκμετάλλευση του περιβάλλοντα χώρου του είτε για σχεδιασμό κίνησης είτε για αναζήτηση αντικειμένων. Η μελέτη τέτοιων διεργασιών μέσω σύγχρονων τεχνολογικών διατάξεων και μεθόδων επεξεργασίας έχει την προοπτική να διαλευκάνει το μυστήριο του εγκεφάλου και να ρίξει φως σε βασικές λειτουργίες του [1].

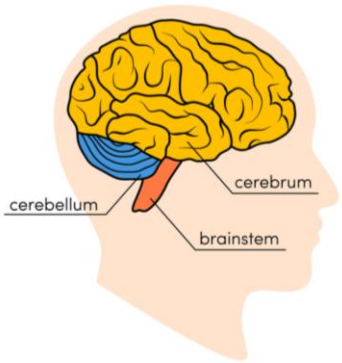
5. Εγκεφαλική Δραστηριότητα – Θεωρητικό Υπόβαθρο

5.1. Λειτουργικός Διαχωρισμός Εγκεφάλου

Ο εγκέφαλος είναι μία από τις πιο περίπλοκες βιολογικές δομές που γνωρίζουμε. Κυρίως τα τελευταία χρόνια γίνονται συνεχείς προσπάθειες κατανόησης σε βάθος των δομών και των λειτουργιών του, ωστόσο τα αναπάντητα ερωτήματα συνεχίζουν να υπερτερούν έναντι των πλήρως διερευνημένων.

Ο εγκέφαλος είναι το βασικότερο όργανο του Κεντρικού Νευρικού Συστήματος (ΚΝΣ), το οποίο λαμβάνει και επεξεργάζεται πληροφορίες από τα αισθητήρια όργανα του σώματος, ρυθμίζει πλήθος ζωτικών και μη λειτουργιών του και ελέγχει ανώτερες νοητικές διεργασίες [1].

Ο ανθρώπινος εγκέφαλος βρίσκεται εντός του κρανίου, περιβάλλεται από τρείς προστατευτικούς υμένες, τις μήνιγγες, και αποτελείται από τον «τελικό» εγκέφαλο (cerebrum), την παρεγκεφαλίδα (cerebellum) και το εγκεφαλικό στέλεχος (brainstem).



Εικόνα 1 *Διάγραμμα του ανθρώπινου εγκεφάλου, που δείχνει την παρεγκεφαλίδα (cerebellum), το εγκεφαλικό στέλεχος (brainstem) και τον <τελικό> εγκέφαλο (cerebrum)*

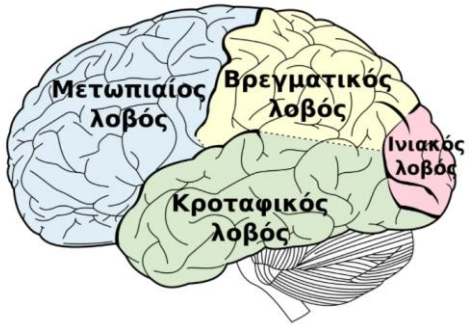
Ο εγκεφαλικός φλοιός (cerebral cortex) κατέχει το μεγαλύτερο μέρος του εγκεφάλου. Λόγω του χαρακτηριστικού γκρι χρώματος που έχει το εξωτερικό των διατηρημένων εγκεφάλων, ονομάζεται και φαιά ουσία. Η επιφάνεια του εγκεφαλικού φλοιού έχει ογκώματα και σχισμές, γνωστά ως έλικες και αύλακες αντίστοιχα, με αποτέλεσμα περίπου τα 2/3 του φλοιού να βρίσκονται κρυμμένα [2]. Η χωροταξία των ελίκων και των αυλάκων είναι παρεμφερής, ωστόσο μπορεί να διαφοροποιείται μεταξύ των ανθρώπων.

Κατά μήκος του εγκεφάλου υπάρχει η μεγαλύτερη σχισμή του, η επιμήκης σχισμή (longitudinal fissure), η οποία τον χωρίζει σε δύο τμήματα τα γνωστά και ως ημισφαίρια. Τα ημισφαίρια ενώνονται μέσω του μεσολοβίου (corpus callosum), μίας πυκνής δέσμης νευραξόνων που επιτρέπει την ανταλλαγή πληροφοριών μεταξύ των δύο πλευρών. Οι δύο πλευρές δεν είναι απαραίτητα συμμετρικές και η απώλεια συμμετρίας τους δεν είναι μόνο δομική αλλά γενικεύεται και στη λειτουργία τους, με το καθένα να περιλαμβάνει κέντρα διαφορετικών δεξιοτήτων και αντιληπτικών δυνατοτήτων. Λειτουργίες λογικής όπως είναι η γλώσσα, η γραφή, η ανάλυση και η ομιλία βρίσκονται στο αριστερό ημισφαίριο κυρίως, ενώ στο δεξί ανήκει η αντίληψη του χώρου και του χρόνου, η ακουστική και οπτική επίγνωση, τα αισθήματα, η φαντασία και γενικά οι δημιουργικές ικανότητες. Κάθε ημισφαίριο ελέγχει την αντίθετη σε αυτό πλευρά του σώματος, για παράδειγμα το αριστερό ημισφαίριο δέχεται ερεθίσματα από τη δεξιά πλευρά του σώματος και αντίστροφα.

Ο φλοιός διαιρείται σε διαφορετικές περιοχές χάρη στους έλικες και στις αύλακες βάσει είτε των δομικών είτε των λειτουργικών χαρακτηριστικών τους, όμως τα όρια αυτά δεν είναι πάντα σαφή και η επικάλυψη αυτών είναι πιθανή.

Ο κύριος διαχωρισμός που στηρίζεται σε ανατομικά χαρακτηριστικά είναι η κατάταξη σε τέσσερα ζεύγη λοβών [3], των οποίων το μέγεθος και η ονομασία σχετίζονται άμεσα με τα οστά του κρανίου που βρίσκονται από πάνω τους.

- Ο μετωπιαίος λοβός (frontal lobe) βρίσκεται κάτω από το μετωπιαίο οστό, δηλαδή στο μπροστινό μέρος του εγκεφάλου. Σχετίζεται με τη λύση προβλημάτων, την κίνηση, καθώς ο κινητικός φλοιός βρίσκεται εκεί, την εκφραστική γλώσσα και εμπλέκεται γενικά σε νοητικές λειτουργίες που θεωρούνται ανώτερες.
- Ο βρεγματικός λοβός (parietal lobe) βρίσκεται κάτω από το βρεγματικό οστό στο μέσο του εγκεφάλου. Σχετίζεται με απτικές αισθητηριακές πληροφορίες όπως η πίεση, η αφή και ο πόνος και είναι επίσης έδρα του σωματοαισθητικού φλοιού (somatosensory cortex), ο οποίος είναι σημαντικός για τη λειτουργία των αισθήσεων.
- Ο κροταφικός λοβός (temporal lobe) βρίσκεται κάτω από το κροταφικό οστό στα πλαϊνά μέρη του εγκεφάλου. Σχετίζεται με τη ακοή και την αναγνώριση των ήχων αφού εκεί βρίσκεται ο πρωτεύων ακουστικός φλοιός. Επίσης περιλαμβάνει τον ιππόκαμπο, ο οποίος συσχετίζεται με τη δημιουργία νέων αναμνήσεων και την εκμάθηση νέων πραγμάτων. Για αυτόν το λόγο ο κροταφικός λοβός έχει στενή σχέση με την ακοή και τη μνήμη.
- Ο ινιακός λοβός (occipital lobe) βρίσκεται κάτω από το ινιακό οστό στο πίσω μέρος του εγκεφάλου. Σχετίζεται με την ερμηνεία των οπτικών ερεθισμάτων και πληροφοριών, καθώς εκεί εδρεύει ο πρωτεύων οπτικός φλοιός (primary visual cortex).



Εικόνα 2 *Απεικόνιση των κύριων λοβών του ανθρώπινου εγκεφάλου*

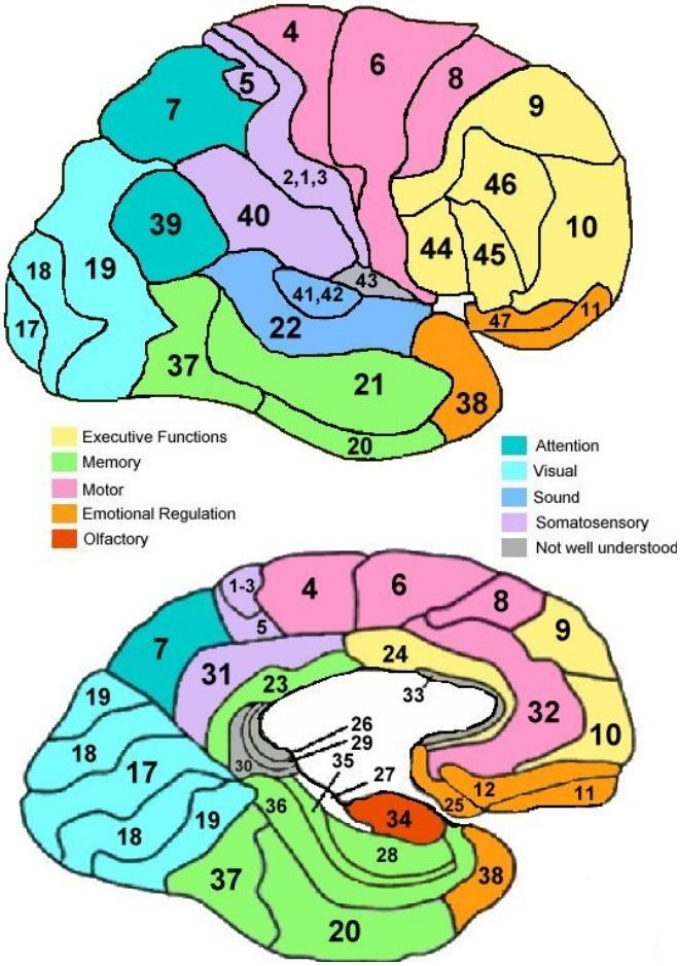
Όσον αφορά τη λειτουργικότητα του εγκεφάλου, υπάρχουν δύο κατηγορίες λειτουργικών περιοχών: οι αισθητηριακές (sensory) και οι κινητικές (motor).

Για κάθε μία από τις πέντε βασικές αισθήσεις, όραση, ακοή, γεύση, αφή και όσφρηση, υπάρχει ένας πρωτεύων (primary cortex) και ένας συνειρμικός (association cortex) αισθητηριακός φλοιός. Ο πρωτεύων προσφέρει την επίγνωση της ίδιας της αίσθησης, ενώ ο συνειρμικός αποδίδει νόημα στην αίσθηση και τη συσχετίζει με άλλες πληροφορίες.

Στον έλεγχο των εκούσιων κινήσεων, κυρίως των λεπτών κινήσεων των χεριών, εμπλέκονται οι κινητικές περιοχές, οι οποίες περιλαμβάνουν τον πρωτεύοντα κινητικό φλοιό (primary motor cortex), τον προ-κινητικό φλοιό (premotor cortex), το μετωπιαίο πεδίο των ματιών (frontal eye field), και την περιοχή Broca (Broca’s area) [4]. Οι κινητικές περιοχές εκτείνονται από το ένα αυτί ως το άλλο σαν ακουστικά. Οι δεξιές περιοχές ελέγχουν την αριστερή πλευρά του σώματος και αντίστροφα [5].

Ο χάρτης Brodmann (Brodmann map) είναι ένας ακόμα γνωστός διαχωρισμός του εγκεφαλικού φλοιού, ο οποίος περιέχει μεγαλύτερη ακρίβεια. Ο χάρτης αυτός δημοσιεύτηκε για πρώτη φορά το 1909 από τον Γερμανό ανατόμο Korbinian Brodmann και βασίζεται στην κυτταροαρχιτεκτονική,

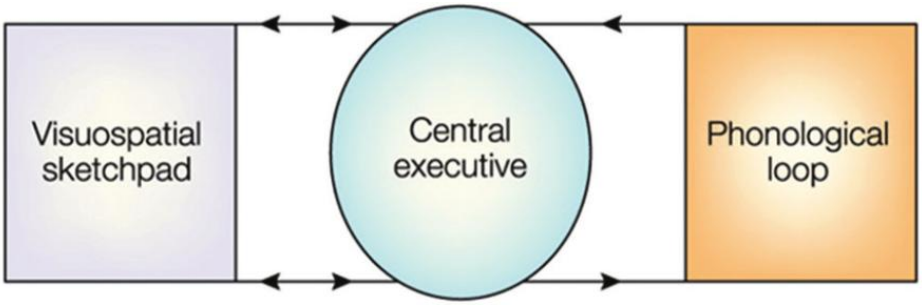
δηλαδή την ιστολογική δομή και οργάνωση των κυττάρων. Οι περιοχές διαχωρίζονται με βάση την ιστολογική ομοιογένειά τους και ενώνονται σε ζώνες (regions) όταν έχουν παρόμοια σύνθεση. Ο χάρτης διακρίνει τον εγκέφαλο σε 43 περιοχές (αριθμημένες με αριθμούς μεταξύ 1 και 52, με τους αριθμούς όμως 12-16 και 48-51 να απουσιάζουν) και 11 ζώνες [6]. Η οριοθέτησή του δεν είναι απόλυτη και τα όριά του δεν έχουν πλήρη σαφήνεια, καθώς κάθε εγκέφαλος παρουσιάζει διαφορετικές τοπογραφικές δομές σε σχέση με το μοντέλο αναφοράς που χρησιμοποιήθηκε από τον Brodmann.



Εικόνα 3 Χάρτης *των περιοχών του εγκεφάλου σύμφωνα με τον χάρτη του Brodmann*

5.2. Χωρική Αναπαράσταση & Χωρικός Προσανατολισμός

Ο μηχανισμός με τον οποίο οι άνθρωποι συγκρατούν για σύντομο χρονικό διάστημα πληροφορίες με σκοπό να τις επεξεργαστούν ονομάζεται Εργαζόμενη Μνήμη (Working Memory). Η εργαζόμενη μνήμη αποτελείται από ένα κεντρικό στέλεχος και δύο αποθηκευτικά υποσυστήματα, τον λεκτικό βρόγχο (verbal loop) και το οπτικο-χωρικό μπλοκ (visuo-spatial sketchpad), όπου κρατείται ενεργός περιορισμένος αριθμός λεκτικών και οπτικο-χωρικών πληροφοριών, αντίστοιχα [7].



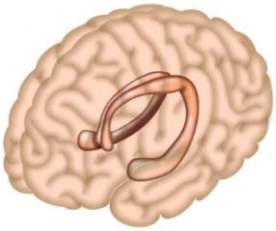
Εικόνα 4 *Απεικόνιση της Εργαζόμενης Μνήμης*

Για τη διατήρηση πληροφοριών στην εργαζόμενη μνήμη για παρατεταμένο χρονικό διάστημα φαίνεται να παίζουν ρόλο και τα δύο ημισφαίρια, όμως το δεξί φαίνεται να είναι πιο εξειδικευμένο [7]. Οι περιοχές που ενεργοποιούνται κατά τη λειτουργία της χωρικής εργαζόμενης μνήμης είναι ο πρωτεύων ενδορρινικός φλοιός (entorhinal cortex - EC[1], περιοχές Brodmann 28 και 34), ο οπισθοσπληνοειδής φλοιός (retrosplenial cortex, περιοχές Brodmann 29 και 30), ο δεξής ραχιοπλευρικός προμετωπιαίος φλοιός (dorsolateral prefrontal cortex – DLPFC, πλάγιο τμήμα της περιοχής 9 και περιοχή 46 Brodmann), ο δεξιός οπίσθιος βρεγματικός φλοιός (posterior parietal cortex - PPC) και ο ιππόκαμπος [8]. Μελέτες έχουν δείξει τη σημαντικότητα του ιππόκαμπου στη διατήρηση των χωρικών πληροφοριών για μεγάλο χρονικό διάστημα, ενώ για μικρή χρονική περίοδο φαίνεται να έχουν ιδιαίτερη σημασία ο δεξιός ραχιοπλευρικός μετωπιαίος και ο δεξιός οπίσθιος βρεγματικός φλοιός [9].

Για να μπορέσουν να αξιοποιηθούν οι χωρικές πληροφορίες της εργαζόμενης μνήμης, όμως, και να μπορεί ο άνθρωπος να προσαρμόζεται σε νέα περιβάλλοντα και να πηγαίνει από ένα σημείο σε ένα άλλο χρειάζεται την ικανότητα του Χωρικού Προσανατολισμού (Spatial Orientation). Ο χωρικός προσανατολισμός είναι η ικανότητα διατήρησης του προσανατολισμού και της στάσης του σώματος σε σχέση με το περιβάλλον σε κατάσταση ηρεμίας και κίνησης [10]. Για να επιτευχθεί αυτό, απαιτείται μια δυναμική ενημέρωση της αναπαράστασης των σχέσεων μεταξύ του σώματος και του περιβάλλοντος. Η ενημέρωση αυτή, εξαρτάται από την κεντρική ενσωμάτωση των πολυαισθητηριακών πληροφοριών της τρέχουσας στιγμής αλλά και από τη σύγκριση των αισθητηριακών σημάτων με τις προγραμματισμένες από πριν τροχιές, το σχήμα του σώματος και τις προηγούμενες αναμνήσεις. Τρεις κύριες αισθητηριακές μέθοδοι εμπλέκονται σε αυτήν τη διαδικασία, η όραση, το αιθουσαίο σύστημα (vestibular system - σύστημα ισορροπίας) και η ιδιοδεκτικότητα (proprioception, γνωστή και ως κιναισθησία - kinesthesia) [11].

Πιο συγκεκριμένα, ιδιοδεκτικότητα ονομάζεται η ικανότητα του σώματος να αισθανθεί την κίνηση, τη δράση και την τοποθεσία. Δίνει τη δυνατότητα στα άτομα να κινούνται χωρίς να σκέφτονται ακριβώς τις επόμενες κινήσεις τους συνειδητά, όπως για παράδειγμα το περπάτημα (το άτομο δεν σκέφτεται συνειδητά πού θα τοποθετήσει το πόδι του στη συνέχεια). Η ιδιοδεκτικότητα είναι αποτέλεσμα αισθητηριακών υποδοχέων που εδράζονται κυρίως στους μύες, τις αρθρώσεις και τους τένοντες του σώματος, που όταν κινούνται στέλνουν λεπτομερή μηνύματα στον εγκέφαλο σχετικά με τη θέση και την ενέργειά τους. Ο εγκέφαλος, στη συνέχεια, επεξεργάζεται αυτά τα μηνύματα και σε συνεργασία με την όραση και το αιθουσαίο σύστημα δημιουργεί την αντίληψη για τη θέση και την κίνηση του σώματος [12].

Η εξειδικευμένη περιοχή του εγκεφάλου για την πλοήγηση σε χωρικό περιβάλλον είναι ο ιππόκαμπος. Ο ιππόκαμπος βοηθάει τα άτομα να προσδιορίσουν το πού βρίσκονται, πώς έφτασαν στο συγκεκριμένο σημείο και πώς να πλοηγηθούν στον επόμενο προορισμό τους. Η χρήση ικανοτήτων προσανατολισμού και πλοήγησης μπορούν να προκαλέσουν την ανάπτυξη του ιππόκαμπου και του εγκεφάλου σχηματίζοντας περισσότερες νευρικές οδούς [13].



Εικόνα 5 *Απεικόνιση του ιππόκαμπου*

[1] https://en.wikipedia.org/wiki/Entorhinal_cortex

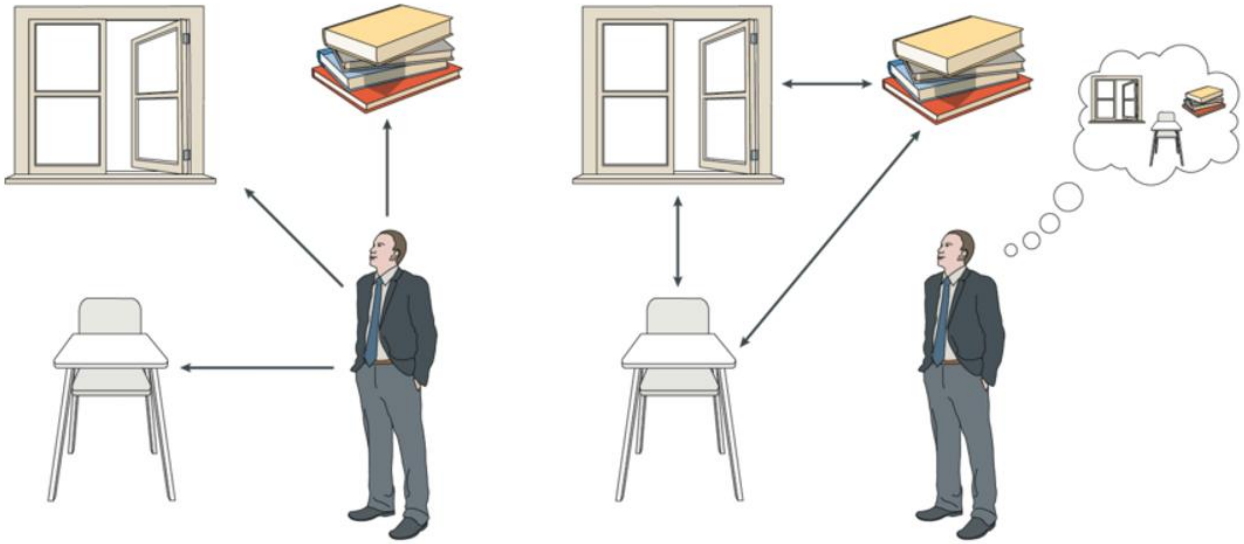
Τη διαδικασία του χωρικού προσανατολισμού βοηθούν σημαντικά τα ορόσημα. Ως ορόσημο ορίζεται ένα ακίνητο αντικείμενο σε ένα περιβάλλον, το οποίο λόγω της σταθερότητάς του μπορεί να χρησιμοποιηθεί σαν ένδειξη ή σημείο αναφοράς. Το 1981 ο Evans και οι συνεργάτες του [14] έδειξαν ότι, όταν εισάγεται σε ένα νέο περιβάλλον, ένα άτομο μαθαίνει ένα σύνολο από ορόσημα μέσα στις πρώτες δύο εβδομάδες διαμονής του και ο αριθμός αυτός δεν αυξάνεται κατά τη διάρκεια ενός έτους. Επίσης οι αποφάσεις σχετικά με τον προσανατολισμό γίνονται πιο γρήγορα όταν οι άνθρωποι φαντάζονται τον εαυτό τους σε σημεία αναφοράς, και η απόσταση από ένα ουδέτερο σημείο σε ένα σημείο αναφοράς κρίνεται πιο γρήγορα από ότι η απόσταση σε μη σημείο αναφοράς [14].

Συνεπώς, για την επίτευξη του χωρικού προσανατολισμού χρειάζεται η ικανότητα της επίγνωσης και αντίληψης των χωρικών σχέσεων εαυτού-περιβάλλοντος αλλά και του ίδιου του εαυτού. Η ικανότητα αυτή ονομάζεται Χωρική Αντίληψη (Spatial Perception). Η χωρική ευαισθητοποίηση αποτελείται από την εξωδεκτική (exteroceptives) διαδικασία, που δημιουργεί αναπαραστάσεις για τον χώρο μέσω αισθήσεων και την ενδοδεκτική (interoceptive), που δημιουργεί αναπαραστάσεις για το σώμα όπως η θέση και ο προσανατολισμός [15].

Επιπλέον, στη χωρική αντίληψη και μνήμη κεντρική σημασία έχει η έννοια του πλαισίου αναφοράς. Πιο αναλυτικά, ένα πλαίσιο αναφοράς καθορίζει τί είδους χωρική τοποθεσία εκπροσωπείται και πώς καθορίζονται οι θέσεις σε μία χωρική αναπαράσταση. Οι άνθρωποι και κάποια ζώα χρησιμοποιούν μεγάλο πλήθος από πλαίσια, ωστόσο τα τελευταία χρόνια πιστεύεται ότι χρησιμοποιούνται κυρίως δύο βασικές κατηγορίες: το εγωκεντρικό (egocentric) και το αλλοκεντρικό (allocentric) [16].

Αρχικά, η επεξεργασία των αισθητηριακών εισόδων παράγει αναπαραστάσεις στους πρωτογενείς αισθητηριακούς φλοιούς που είναι συγκεκριμένες για κάθε αίσθηση, όπως ρετινοτοπικές (retinotopic) στην όραση και σωματοτοπικές (somatotopic) στην αφή (στην ακοή ο πρωτογενής αισθητηριακός φλοιός έχει τονοτοπική (tonotopic) μη χωρική αναπαράσταση). Η ενσωμάτωση οπτικών, ακουστικών και σωματοαισθητηριακών πληροφοριών με σήματα (π.χ. θέση ματιών) που σχετίζονται με τη θέση του σώματος και των μερών του σώματος στον χώρο συντελεί στις δύο αυτές θεμελιώδης κατηγορίες αναπαραστάσεων αντικειμένων αλλά και του σώματος στο χώρο. Στα εγωκεντρικά πλαίσια, η θέση των αντικειμένων κωδικοποιείται με αναφορά ολόκληρο το σώμα ή μέρη αυτού (π.χ. το χέρι), δημιουργώντας αναπαραστάσεις οι οποίες μπορεί να είναι κεντραρισμένες στο κεφάλι (στον οπτικό τομέα ως αποτέλεσμα της σύνθεσης του ρετινοτοπικού χάρτη και της θέσης των ματιών), κεντραρισμένες στον κορμό (ως αποτέλεσμα πληροφοριών για τη θέση του κεφαλιού και τη στάση του σώματος), κεντραρισμένες στο χέρι και ούτω καθεξής. Στα αλλοκεντρικά πλαίσια, τα αντικείμενα κωδικοποιούνται κυρίως με βάση τις χωρικές τους ιδιότητες και τη διαμόρφωσή τους, όπως οι σχέσεις μεταξύ των μερών τους και μεταξύ άλλων αντικειμένων που βρίσκονται στον χώρο [17].

Οι εγωκεντρικές αναπαραστάσεις μπορούν να χρησιμοποιηθούν στην οργάνωση κινήσεων για την επίτευξη ενός στόχου ή για την αποφυγή επιβλαβών ερεθισμάτων. Από την άλλη μεριά, οι αλλοκεντρικές αναπαραστάσεις μπορούν να χρησιμοποιηθούν για τον εντοπισμό αντικειμένων και την πλοήγηση στον χώρο. Στην αλλοκεντρική κατάσταση, πέρα από τον κοντινό χώρο δράσης, ο μακρινός χώρος είναι ουσιαστικά ένας καθαρά οπτικός χώρος που φαινομενολογικά γίνεται αντιληπτός ως μη Ευκλείδειος, με μη-γραμμική κλιμάκωση της απόστασης [17].



Εικόνα 6 *Σχεδιάγραμμα διαφορών μεταξύ εγωκεντρικής και αλλοκεντρικής αναπαράστασης*

Όσον αφορά τα απτικά ερεθίσματα, έχει θεωρηθεί ότι η αντίληψη ενός απτικού ερεθίσματος μπορεί να καθοριστεί όχι μόνο από τους σωματοαισθητικούς παράγοντες, αλλά και από τη χωρική θέση του δέρματος που αγγίζεται, με αναφορά σε εγωκεντρικά πλαίσια συντεταγμένων, με επίκεντρο τα μεμονωμένα μέρη του σώματος. Αυτή η διασταύρωση μεταξύ διαφορετικών συστημάτων συντεταγμένων επιτρέπει την παρακολούθηση των διαφόρων μελών του σώματος στο χώρο και κάθε μεμονωμένου ερεθίσματος που έρχεται σε επαφή με αυτά, υποστηρίζοντας μεγάλο βαθμό ευελιξίας κατά τη διάρκεια των κινήσεων [17].

Δεδομένα συμπεριφοράς υποδηλώνουν ότι οι άνθρωποι εναλλάσσουν αυθόρμητα μεταξύ των δύο αυτών κύριων κατηγοριών πλαισίων, ή σε μερικές περιπτώσεις μπορεί αυτά ακόμα και να συνυπάρχουν και να λειτουργούν παράλληλα. Επιπλέον, έχει προταθεί ότι η χωρική μνήμη υποστηρίζει ως επί το πλείστον εγωκεντρικά πλαίσια αναφοράς, ενώ υπάρχουν ισχυρές ενδείξεις ότι οι αλλοκεντρικές αναπαραστάσεις υποστηρίζονται σε επίπεδο νευρώνα [16].

Όσον αφορά τα μέρη του εγκεφάλου που ενεργοποιούνται κατά τη χρήση του εκάστοτε τύπου πλαισίου, έρευνες έχουν δείξει ότι στην εγωκεντρική κατάσταση ενεργοποιούνται ο οπίσθιος βρεγματικός λοβός, η μετωπιαία έλικα και ο κινητικός φλοιός διμερώς, αν και η ενεργοποίηση είναι πιο εκτεταμένη στο δεξί ημισφαίριο [16]. Στη αλλοκεντρική κατάσταση ενεργοποιείται πάλι ένα μετωπιαίο δίκτυο επικεντρωμένο στις άνω βρεγματικές περιοχές (superior parietal regions) και στον ανώτερο μετωπιαίο φλοιό του δεξιού ημισφαιρίου, ωστόσο η αλλοκεντρική χωρική κωδικοποίηση γίνεται κυρίως στον έσω κροταφικό λοβό (MTL) που περιλαμβάνει τον ιππόκαμπο, την αμυγδαλή και παραϊποκαμπικές περιοχές [18].

Έχει παρατηρηθεί ότι τα βρεγματικά και μετωπιαία δίκτυα ενεργοποιούνται και στις δύο κύριες κατηγορίες πλαισίων αλλά πιο ισχυρά στα εγωκεντρικά. [16]

5.3. Συνδρομή Όρασης & Λειτουργία υπό Απουσία Οπτικών Ερεθισμάτων

Η όραση είναι μία από τις πέντε αισθήσεις του ανθρώπου και από πολλούς θεωρείται η σημαντικότερη, καθώς μέσω αυτής γίνεται επί το πλείστον αντιληπτό το εξωτερικό περιβάλλον. Η

σημασία αυτή υπερτονίζεται κιόλας από το γεγονός ότι περίπου το 30% του ανθρώπινου εγκεφάλου ασχολείται με την επεξεργασία και ερμηνεία των οπτικών ερεθισμάτων [19].

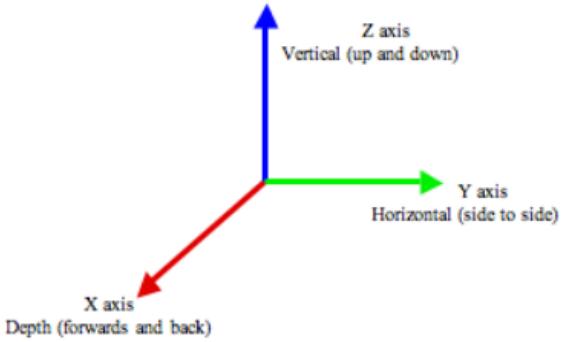
Η απώλεια όρασης, λοιπόν, σε βαθμό που δεν μπορεί να επιδιορθωθεί με τεχνητά μέσα όπως οι φακοί, μπορεί να ορθώσει πολλά εμπόδια στη ζωή ενός ανθρώπου. Ωστόσο, το γεγονός ότι τα άτομα με τύφλωση μπορούν να λύσουν χωρικά προβλήματα και ότι εκείνα με πλήρη όραση κάνουν λάθη [20], σημαίνει ότι η όραση δεν είναι ούτε απαραίτητη ούτε αρκετή για τη χωρική κωδικοποίηση. Υπάρχουν δύο αντικρουόμενες θεωρίες που έχουν προκύψει από πρώιμα πειραματικά ευρήματα σχετικά με το αν η απώλεια της όρασης συνοδεύεται ή όχι από βελτίωση άλλων αισθήσεων [21].

Από τη μία πλευρά, υποστηρίζεται η υπόθεση του ελλείμματος αντίληψης [21], όπου από την απουσία οπτικής πληροφορίας, τα άτομα μπορεί να αναπτύξουν γνωστικά χωρικά ελλείμματα σε άλλες αισθητηριακές λειτουργίες. Αυτή η υπόθεση υποστηρίχτηκε από μεγάλο όγκο έρευνας πάνω σε ζώα η οποία κατέδειξε τη σημασία της οπτικής ανατροφοδότησης στην ακουστική χωρική εκμάθηση και στη φυσιολογική ανάπτυξη των ακουστικών χωρικών χαρτών στο άνω διδύμιο (superior colliculus) [21].

Από την άλλη πλευρά, υπάρχει η θεωρία της αντιστάθμισης [21], σύμφωνα με την οποία τα τυφλά άτομα αναπτύσσουν εξαιρετικές αντιληπτικές ικανότητες στις υπόλοιπες αισθητηριακές δυνατότητες τους για να αντισταθμίσουν την οπτική απώλεια. Η προϋπάρχουσα υποστήριξη αυτής της θεωρίας προέρχεται, μεταξύ άλλων, από τους Ντενί Ντιντερό στο Γράμμα στους Τυφλούς (1749) [22] και Γουίλιαμ Τζέιμς (1842-1910) [23]. Μεταγενέστερες μελέτες επιβεβαίωναν την υπόθεση της αντιστάθμισης αποδεικνύοντας ανώτερες χωρικές ακουστικές ικανότητες σε πρώιμα τυφλά άτομα [21]. Άλλωστε, οι ακουστικές χωρικές εργασίες έχουν αποδειχθεί ότι προκαλούν σημαντική ενεργοποίηση στον οπτικό φλοιό των πρώιμα τυφλών ατόμων και ξεχωριστές ικανότητες ακουστικού εντοπισμού έχουν αποδειχθεί ότι σχετίζονται εντόνως με το μέγεθος της δραστηριότητας του οπτικού φλοιού [21].

Το φαινόμενο αυτό ονομάζεται νευροπλαστική (neuroplasticity) και είναι η ικανότητα του νευρικού συστήματος να προσαρμόζει τη λειτουργική και δομική του οργάνωση σαν απάντηση στην ανάπτυξη, την εμπειρία και το περιβάλλον. Οι διατροπικές (cross-modal) νευροπλαστικές αλλαγές στον εγκέφαλο συμβαίνουν συχνά μετά την αισθητηριακή στέρηση, με σκοπό οι περιοχές του εγκεφάλου που συνήθως σχετίζονται με την απολεσθείσα αίσθηση να στρατολογούνται για την επεξεργασία πληροφοριών από τις υπόλοιπες αισθητηριακές εισόδους [24]. Για παράδειγμα, στον εκ γενετής τυφλό εγκέφαλο, ο ινιακός φλοιός επεξεργάζεται μη-οπτικά σήματα (σε αντίθεση με τις φυσιολογικές συνθήκες) όπως ακουστικά, γλωσσικά ή απτικά ερεθίσματα [24]. Μάλιστα, υπάρχει ένα αυξανόμενο σώμα ερευνών πάνω στη νευροαπεικόνιση που παρέχει στοιχεία τα οποία υποδηλώνουν ότι οι ενδοφλοιϊκές συνδέσεις (cortico-cortical pathways) μεταξύ του ακουστικού και του οπτικού φλοιού μπορεί να αποτελούν τη βάση για τη διατροπική επεξεργασία [21].

Ωστόσο, παρά την ύπαρξη των δεδομένων που υποστηρίζουν τη θεωρία της αντιστάθμισης, είναι σημαντική μια πιο προσεκτική ματιά στις συνθήκες κάτω από τις οποίες παρατηρείται η βελτίωση των ακουστικών χωρικών ικανοτήτων.



Εικόνα 7 *Διακριτές ζώνες χωρικού περιβάλλοντος*

Αρχικά, το χωρικό περιβάλλον μπορεί να διαιρεθεί σε διακριτές ζώνες. Σε σχέση με τη χωρική ακοή, τυπικά χωρίζεται σε οριζόντια, κατακόρυφα και επίπεδα βάθους [21]. Κατά μέσο όρο, τα τυφλά άτομα μπορούν καλύτερα να εντοπίσουν ήχους στο οριζόντιο επίπεδο, αλλά παρουσιάζουν έλλειμμα στο κατακόρυφο [21]. Ενδιαφέρον, μάλιστα, είναι και το γεγονός ότι στα τυφλά άτομα παρατηρήθηκε μία αντίστροφη συσχέτιση. Πιο αναλυτικά, όσοι παρουσίασαν μεγαλύτερη ακρίβεια στο οριζόντιο πλαίσιο ήταν επίσης αυτοί με την μεγαλύτερη αδυναμία στο κατακόρυφο [21]. Διευκρινίζεται ότι η συσχέτιση αυτή δεν παρατηρείται στα άτομα χωρίς προβλήματα όρασης [21]. Το γεγονός αυτό, όχι μόνο τάσσεται κατά της ιδέας για γενική ακουστική χωρική βελτίωση στα τυφλά άτομα, αλλά επίσης υποδηλώνει την πιθανότητα αντιστάθμισης στην επίδοση εντοπισμού μεταξύ των δύο αυτών επιπέδων.

Όσον αφορά την εκτίμηση σχετικού βάθους βάσει ακουστικής πληροφορίας, τα άτομα με πρώιμη τύφλωση έχει δειχθεί ότι είναι πιο ακριβή από τα άτομα με όραση, ενώ αντίθετα υστερούν στην εκτίμηση απόλυτης απόστασης, δηλαδή απόσταση μεταξύ παρατηρητή και πηγής [21].

Σχετικά με τα άτομα με όψιμη τύφλωση, στοιχεία δείχνουν ότι οι χωρικές ακουστικές ικανότητές τους βρίσκονται κάπου στη μέση μεταξύ των πρώιμα τυφλών ατόμων και των ατόμων με όραση. Πιο αναλυτικά, δεν επωφελούνται από πολλές από τις βελτιώσεις που παρουσιάζουν τα πρώιμα τυφλά άτομα, αλλά ούτε παρουσιάζουν τα ελλείμματα [21].

Εκτός από την ακοή, τα άτομα με τύφλωση χρησιμοποιούν εκτενώς την αφή ως μέσω εξερεύνησης και κωδικοποίησης του χώρου που τους περιβάλλει. Η αφή λειτουργεί με αισθητήρια νεύρα που βρίσκονται κάτω από το δέρμα. Η κατανομή των αισθητήριων οργάνων διαφέρει ανάμεσα στα σημεία του σώματος, με μεγαλύτερη πυκνότητα να υπάρχει στα άκρα και κυρίως στις παλάμες και το κεφάλι [25]. Τα άτομα, και κυρίως αυτά που πάσχουν από απώλεια μίας ή περισσότερων αισθήσεων, μπορούν να χρησιμοποιήσουν τις υπόλοιπες αισθήσεις τους για να προσλάβουν τις χωρικές πληροφορίες που χρειάζονται. Άλλωστε για να αναπαρασταθεί η ίδια πληροφορία μπορούν να χρησιμοποιηθούν πολλές μορφές κωδικοποίησης [26].

Τα χαρακτηριστικά των κινήσεων που χρησιμοποιούνται στην εξερεύνηση ενός χώρου επηρεάζουν την αναπαράσταση και την κρίση του ατόμου για τον συγκεκριμένο χώρο, όπως για παράδειγμα τη σχετική ή απόλυτη τοποθεσία σημείων ή των αποστάσεων μεταξύ τους. Οι κινήσεις παίζουν ρόλο στην κατασκευή των αναπαραστάσεων ιδιαίτερα για εκτίμηση απόστασης τόσο στα άτομα με τύφλωση όσο και στα χωρίς, υποδεικνύοντας ότι οι άξονες αναφοράς παίζουν ρόλο στην αναπαράσταση που χρησιμοποιείται για την εκτίμηση θέσης. [26]

Επίσης, μελέτες υποστηρίζουν ότι η χρήση εξωτερικών σημείων αναφοράς είναι πιθανή. Η ικανότητα αναγνώρισης αναφορών είναι πολύ μειωμένη όταν ο παρατηρητής στερείται όρασης, φυσικά, αλλά είναι ακόμα πιθανή η χρήση διακριτικών χαρακτηριστικών τακτικής, χωρικών αξόνων, ή μερών του

σώματος. Τα εκ γενετής τυφλά άτομα φαίνεται να είναι λιγότερο ικανά στην αξιοποίηση τέτοιων αναφορών από τους μετέπειτα τυφλούς [26].

Όσον αφορά την εκτίμηση της ευκλείδειας απόστασης μεταξύ ενός αρχικού και ενός τελικού σημείου, και οι δύο ομάδες -άτομα με απώλεια όρασης και χωρίς- υπερεκτιμούν την απόσταση όσο αυτή μεγαλώνει. Αυτό το φαινόμενο είναι πιο έντονο στους εκ γενετής τυφλούς παρατηρητές παρά σε αυτούς που έχασαν την όρασή τους αργότερα, προτείνοντας ότι η οπτική εμπειρία βοηθάει στον συμπερασμό της χωρικής απόστασης από την απτική εξερεύνηση [26].

Γενικά, υπάρχει ευελιξία στις ευρετικές στρατηγικές με τις οποίες κωδικοποιούνται πρότυπα από την απτική εξερεύνηση. Στην περίπτωση της εκτίμησης απόστασης, οι παρατηρητές συνήθως χρησιμοποιούν στρατηγικές που βασίζονται στην κίνηση, ενώ στην εκτίμηση θέσης, προτιμούνται οι χωρικοί άξονες. Η τελευταία στρατηγική, μάλιστα, χρησιμοποιείται στην οπτική κωδικοποίηση προτύπων, υποδηλώνοντας ότι όραση και αφή μπορεί να έχουν κοινές ευρετικές κωδικοποίησης. [26]

Το 1997 οι Thinus-Blanc και Gaunet όρισαν τη στρατηγική ως «ένα σύνολο από λειτουργικούς κανόνες που εφαρμόζονται από τον συμμετέχοντα σε διάφορες φάσης της επεξεργασίας πληροφοριών, από την πρώτη κιόλας συνάντηση με μία νέα κατάσταση έως την εξωτερίκευση την χωρικής γνώσης» [27].

Επιπλέον, σημαντικό είναι το ερώτημα εάν τα άτομα με απώλεια όρασης, και κυρίως όσοι είναι είτε εκ γενετής είτε από πολύ μικρή ηλικία τυφλοί, έχουν την δυνατότητα να χρησιμοποιήσουν αλλοκεντρικά πλαίσια αναφοράς.

Τα άτομα με απώλεια όρασης στηρίζονται κυρίως στα εγωκεντρικά πλαίσια αναφοράς για τη διεκπεραίωση χωρικών εργασιών. Τα τυφλά άτομα είναι πιο εξοικειωμένα με σειριακές στρατηγικές στην επεξεργασία πληροφοριών (επειδή η απτική και ακουστική πληροφορία παρέχεται σειριακά) και τείνουν να υιοθετούν σωματοκεντρικά (με βάση το σώμα ως σύνολο) ή χειροκεντρικά (με βάση το χέρι) παρά αλλοκεντρικά συστήματα συντεταγμένων για την οικοδόμηση των εσωτερικών χωρικών αναπαραστάσεων [24]. Η χρήση όμως και αλλοκεντρικών πλαισίων δεν είναι απίθανη. Η έγκαιρη εκπαίδευση παιδιών με συγγενή τύφλωση στον προσανατολισμό και την κινητικότητα, μάλιστα, έχει ως αποτέλεσμα την όξυνση της χωρικής αντίληψης και της αλλοκεντρικής κωδικοποίησης οδηγώντας σε επιδόσεις παρόμοιες με αυτές των ατόμων με κανονική όραση. Όταν χρησιμοποιούνται οι κατάλληλες αλλοκεντρικές στρατηγικές κατά τη διάρκεια της κωδικοποίησης και αναπαράστασης του κοντινού περιβάλλοντος, η απόδοση των ατόμων με τύφλωση μπορεί να εξομοιωθεί με εκείνη των ατόμων με κανονική όραση με κλειστά τα μάτια [27]. Μάλιστα, το 2010 ο Παπαδόπουλος και οι συνεργάτες του [28] μετά το πέρας ενός πειράματος, βρήκαν ότι μόνο περίπου το 1/3 των συμμετεχόντων με τύφλωση χρησιμοποιούσε μόνο εγωκεντρικά πλαίσια για την κωδικοποίηση του άμεσου περιβάλλοντος και μόνο το ¼ των εκ γενετής τυφλών χρησιμοποιούσε στρατηγικές αυτό-αναφοράς.

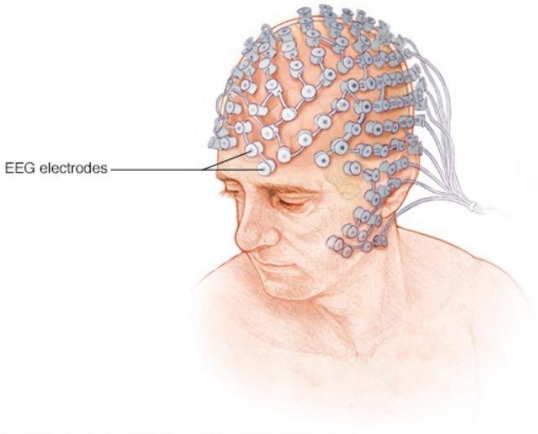
Πίνακας 1 *Χαρακτηριστικά προσανατολισμού ανάλογα με ικανότητα όρασης*

	Τύφλωση εκ γενετής/μικρή ηλικία	**Όψιμη τύφλωση**	**Όραση**
Εντοπισμός ήχων στο οριζόντιο επίπεδο	Βελτιωμένο	Λίγο βελτιωμένο	Μέσο
Εντοπισμός ήχων στο κατακόρυφο επίπεδο	Επιδεινωμένο	Μέσο	Μέσο

Χρήση αναφορών στην αφή	Επιδεινωμένο	Μέσο	Μέσο
Εκτίμηση αυξανόμενης ευκλείδειας απόστασης	Μεγάλη Υπερεκτίμηση	Μικρότερη Υπερεκτίμηση	Μικρότερη Υπερεκτίμηση
Χρήση αλλοκεντρικών πλαισίων	Ναι	Ναι	Ναι

6. Ηλεκτροεγκεφαλογραφία (EEG)

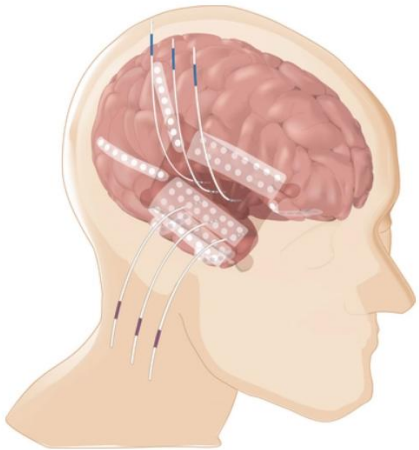
6.1. Εισαγωγή



Εικόνα 8 *Σχέδιο ατόμου κατά την διάρκεια λήψης επιφανειακού ηλεκτροεγκεφαλογραφήματος*

Το ηλεκτροεγκεφαλογράφημα (ΗΕΓ) είναι μία μέθοδος παρακολούθησης και καταγραφής της ηλεκτρικής δραστηριότητας που λαμβάνει μέρος στο τριχωτό της κεφαλής, η οποία έχει αποδειχθεί ότι αντιπροσωπεύει μακροσκοπικά τη δραστηριότητα του επιφανειακού στρώματος του εγκεφάλου. Η ηλεκτρική δραστηριότητα αυτή [29], [30] οφείλεται στη λειτουργία των νευρώνων. Οι νευρώνες είναι κύτταρα που έχουν την ικανότητα να διεγείρονται και να παράγουν ηλεκτρικά και μαγνητικά πεδία όταν αυτό συμβαίνει. Τα πεδία αυτά, στη συνέχεια, είναι δυνατό να καταγραφούν με χρήση ηλεκτροδίων και αναλόγως την τοποθέτηση αυτών η μέθοδος της ηλεκτροεγκεφαλογραφίας χωρίζεται στις εξής βασικές κατηγορίες:

1. Τοπικό εγκεφαλογράφημα (Local EEG): Χρησιμοποιούνται μικροηλεκτρόδια που τοποθετούνται απευθείας στον ιστό, πολύ κοντά στις πηγές της δραστηριότητας, για την καταγραφή των δυναμικών τοπικού πεδίου.
2. Ενδοκρανικό εγκεφαλογράφημα (intracranial EEG – iEEG) ή Φλοιογράφημα (Electrocorticogram - ECoG): Η καταγραφή πραγματοποιείται από την επιφάνεια του εγκεφαλικού φλοιού.
3. Επιφανειακό εγκεφαλογράφημα (EEG): Καταγράφεται επιφανειακά από το τριχωτό της κεφαλής (scalp).



Εικόνα 9 *Απεικόνιση λήψης ενδοκρανιακού εγκεφαλογραφήματος*

Το ηλεκτροεγκεφαλογράφημα, γενικά, έχει πλήθος πλεονεκτημάτων σε σύγκριση με άλλες τεχνικές καταγραφής της εγκεφαλικής δραστηριότητας. Είναι λειτουργικά γρήγορο και έχει υψηλή ακρίβεια στο πεδίο του χρόνου. Με τη σύγχρονη τεχνολογία είναι δυνατή η ανίχνευση της εγκεφαλικής δραστηριότητας σε ανάλυση ενός χιλιοστού του δευτερολέπτου. Συγκεκριμένα για το επιφανειακό εγκεφαλογράφημα, το γεγονός ότι τα ηλεκτρόδια είναι απλώς κολλημένα στο τριχωτό της κεφαλής, κάνει τη διαδικασία μη επεμβατική και άρα κατάλληλη για εύκολη και επαναλαμβανόμενη εφαρμογή τόσο σε υγιή όσο και σε ασθενή άτομα. Τέλος, ο εξοπλισμός του ΗΕΓ είναι σχετικά φθηνός σε σύγκριση με άλλες συσκευές και απλός στη λειτουργία του. Ωστόσο, το βασικότερο μειονέκτημα της συγκεκριμένης τεχνικής είναι η μικρή χωρική ανάλυση της καταγραφής κυρίως σε σχέση με άλλες απεικονιστικές τεχνικές όπως η απεικόνιση μαγνητικού συντονισμού (MRI) και η τομογραφία εκπομπής ποζιτρονίων (PET) [31].

Σε γενικές γραμμές η έρευνα και οι κλινικές εφαρμογές του ηλεκτροεγκεφαλογραφήματος καλύπτουν διάφορους τομείς [30]:

- Μελέτη ιδιοτήτων των εγκεφαλικών και νευρωνικών δικτύων στις νευροεπιστήμες
- Παρακολούθηση του επιπέδου εγρήγορσης και αξιολόγηση της επίδρασης αναισθητικών φαρμάκων
- Διερεύνηση νευρολογικών παθήσεων, όπως η επιληψία, με στόχο την πρόβλεψη, ανίχνευση και εντοπισμό επιληπτικών κρίσεων, καθώς και παθήσεων όπως η άνοια, συμπεριλαμβανομένης της αξιολόγησης των επιδράσεων θεραπευτικών παρεμβάσεων (π.χ., οφέλη και παρενέργειες φαρμακευτικής αγωγής)
- Μελέτη γνωστικών διεργασιών και καταστάσεων, όπως η συγκέντρωση, η κόπωση ή η γνωσιακή απόδοση κατά την εκτέλεση διαφόρων εργασιών
- Χρήσεις νευροανατροφοδότησης EEG ή η βιοανάδρασης EEG πάνω στη θεραπεία φυσιολογικών διαταραχών και νευρολογικών διαταραχών όπως η επιληψία

Τα εγκεφαλικά κύματα και ο τρόπος που αυτά επαναλαμβάνονται συνδέονται μοναδικά με τον κάθε άνθρωπο, γεγονός που συχνά επιτρέπει την αναγνώριση της ταυτότητας ενός ατόμου αποκλειστικά βάσει της εγκεφαλικής του δραστηριότητας [30].

6.2. Φυσιολογία Δημιουργίας Σήματος

Το εγκεφαλογραφικό σήμα στηρίζεται στην αρχή του μεμβρανικού δυναμικού (membrane potential) [32], το οποίο εκφράζει τη διαφορά δυναμικού ανάμεσα στο εσωτερικό και το εξωτερικό της κυτταρικής μεμβράνης:

$$V_m = V_{in} - V_{out}$$

Σε κατάσταση ηρεμίας, η τιμή του δυναμικού είναι συνήθως αρνητική, καθώς το εξωτερικό περιβάλλον της μεμβράνης βρίσκεται σε υψηλότερο δυναμικό σε σχέση με το εσωτερικό. Ως σημείο αναφοράς χρησιμοποιείται συνήθως το εξωτερικό δυναμικό. Για έναν τυπικό νευρώνα, το δυναμικό ηρεμίας είναι περίπου -70mV, αλλά μπορεί να μεταβληθεί σημαντικά.

Η τιμή του μεμβρανικού δυναμικού διαμορφώνεται από την άνιση κατανομή ιόντων ανάμεσα στο εσωτερικό (ενδοκυτταρικό) και το εξωτερικό (εξωκυτταρικό) περιβάλλον του κυττάρου. Αυτή η κατανομή καθορίζεται από τη λειτουργία εξειδικευμένων ιοντικών καναλιών (κυρίως K^+, Na^+ και Cl^-), τα οποία είναι επιλεκτικά για συγκεκριμένους τύπους ιόντων και λειτουργούν ως πηγές ιοντικών

ρευμάτων που ελέγχονται από την τάση. Δύο συνιστώσες αθροίζουν αυτήν την τάση, μία χημική και μία ηλεκτρική, που στο σύνολό τους ορίζουν την ηλεκτροχημική αγωγιμότητα:

- Χημική αγωγιμότητα: Τα ιόντα μετακινούνται μέσω της κυτταρικής μεμβράνης λόγω της διαφοράς συγκέντρωσής τους ανάμεσα στο ενδοκυτταρικό και το εξωκυτταρικό περιβάλλον, μια διαδικασία που ονομάζεται χημική βαθμίδα ή βαθμίδα συγκέντρωσης. Αυτή η μετακίνηση στοχεύει στην εξισορρόπηση των συγκεντρώσεων του κάθε τύπου ιόντων στις δύο πλευρές της μεμβράνης.
- Ηλεκτρική αγωγιμότητα: Κάθε φορά που ένα ιόν διαπερνά τη μεμβράνη λόγω της χημικής βαθμίδας, προκαλεί αλλαγή στη διαφορά δυναμικού ανάμεσα στις δύο πλευρές της μεμβράνης, εξαιτίας της πόλωσής του. Για παράδειγμα, όταν ένα ιόν καλίου μετακινείται προς το εξωκυτταρικό περιβάλλον, αυξάνει το θετικό φορτίο εκτός της μεμβράνης, ενώ μειώνει αντίστοιχα το θετικό φορτίο εντός αυτής. Αυτή η διαδικασία δημιουργεί μια ηλεκτρική βαθμίδα, η οποία εκφράζεται ως διαφορά δυναμικού. Η μεταφορά ιόντων συνεχίζεται όσο υπάρχει διαφορά συγκέντρωσης, με αποτέλεσμα η διαφοροποίηση των φορτίων να εντείνεται. Η διαδικασία αυτή συνεχίζεται έως ότου η ηλεκτρική βαθμίδα αντισταθμίσει πλήρως τη χημική βαθμίδα. Στο σημείο αυτό επιτυγχάνεται ηλεκτροχημική ισορροπία, όπου οι δύο βαθμίδες εξισορροπούνται και δεν παρατηρείται περαιτέρω καθαρή μετακίνηση ιόντων.

Το συνολικό δυναμικό διαμορφώνεται χάρη στην ύπαρξη πολλών καναλιών που διαθέτει η μεμβράνη. Κάθε ένα από αυτά προσπαθεί να πετύχει ηλεκτροχημική ισορροπία για ένα τύπο ιόντος. Αν παρθεί ως παράδειγμα ένα κανάλι ιόντων καλίου, η χημική βαθμίδα ορίζεται ως:

$$\Delta G_{chemical} = R \cdot T \cdot ln\frac{[K^+]_o}{[K^+]_i}$$

Και η ηλεκτρική βαθμίδα ορίζεται ως:

$$\Delta G_{electrical} = z \cdot F \cdot V$$

όπου

$\Delta G_{chemical}$: χημική βαθμίδα

$\Delta G_{electrical}$: ηλεκτρική βαθμίδα

R: παγκόσμια σταθερά αερίων που ισούται με $8.314\ J \cdot K^{-1} \cdot mol^{-1}$

T: θερμοκρασία σε Kelvin

$[K^+]_o$, $[K^+]_i$: εξωκυτταρική και ενδοκυτταρική αντίστοιχα συγκέντρωση των ιόντων καλίου

z: σθένος ιόντος

F: σταθερά Faraday, ισούται με $96.485\ C \cdot mol^{-1}$

V: διαφορά δυναμικού μεμβράνης

Όταν επιτυγχάνεται εξισορρόπηση στις δύο βαθμίδες προκύπτει η εξίσωση του δυναμικού ισορροπίας ή εξίσωση Nerst, προς τιμήν του Walther Nerst που την διατύπωσε. Συγκεκριμένα η εξίσωση πάλι για τα ιόντα καλίου έχει τη μορφή [32]:

$$V_{eq} = \frac{R \cdot T}{z \cdot F} \cdot ln\frac{[K^+]_o}{[K^+]_i}$$

Ωστόσο στη διαμόρφωση του συνολικού μεμβρανικού δυναμικού εκτός από τα ιόντα καλίου, συνήθως συνεισφέρουν και τα ιόντα νατρίου και χλωρίου. Για την ενσωμάτωση του συνόλου των ιόντων διατυπώθηκε η γενίκευση της εξίσωσης Nerst, η εξίσωση Goldman-Hodgkin-Katz (GHK). Για τη συμμετοχή των τριών προαναφερθέντων ιόντων η εξίσωση GHK διαμορφώνεται ως εξής:

$$V_m = \frac{R \cdot T}{V} ln \frac{(p_K \cdot [K^+]_o + p_{Na} \cdot [Na^+]_o + p_{Cl} \cdot [Cl^-]_i)}{(p_K \cdot [K^+]_i + p_{Na} \cdot [Na^+]_i + p_{Cl} \cdot [Cl^-]_o)}$$

Η παράμετρος p είναι διαφορετική για κάθε τύπο ιόντων και εκφράζει τη σχετική διαπερατότητα της μεμβράνης. Για έναν τυπικό νευρώνα σε κατάσταση ηρεμίας ισχύει $p_K = 1$, $p_{Na} = 0.05$, $p_{Cl} = 0.45$ και η προσθήκη αυτών των τιμών διαπερατότητας είναι η κύρια διαφοροποίηση μεταξύ των εξισώσεων Nerst και GHK. Άξιο αναφοράς είναι, επίσης, το γεγονός ότι για τα ιόντα χλωρίου, η εσωτερική συγκέντρωση μπαίνει στον αριθμητή, αντίθετα με τα ιόντα νατρίου και καλίου, διότι είναι ανιόντα –έναντι κατιόντων-.

Στο συνολικό δυναμικό συμβάλλουν πολλοί τύποι ιόντων και για τον λόγο αυτόν, είναι λογικό να υποθέσουμε ότι, σε κάθε στιγμή, κανένα από αυτά τα ιόντα δεν βρίσκεται σε πλήρη ηλεκτροχημική ισορροπία. Ως εκ τούτου, για κάθε τύπο ιόντος, υπάρχει συνεχώς μια ηλεκτροχημική βαθμίδα. Αυτή η βαθμίδα περιγράφεται από την ηλεκτροχημική οδηγό δύναμη (electrochemical driving force - EDF), η οποία για κάθε ιόν X ορίζεται ως εξής:

$$V_{EDF_X} = V_m - V_{eq_X}$$

Το πρόσημο της ηλεκτροχημικής δύναμης σε συνδιασμό με του σθένους του ιόντος (ανιόν ή κατιόν) καθορίζει την πορεία της κίνησης των ιόντων σε σχέση με την κυτταρική μεμβράνη. Ένα θετικό (ή αρνητικό) πρόσημο για ένα κατιόν (ή ανιόν) υποδεικνύει ότι τα ιόντα ρέουν προς το εξωτερικό περιβάλλον της μεμβράνης.

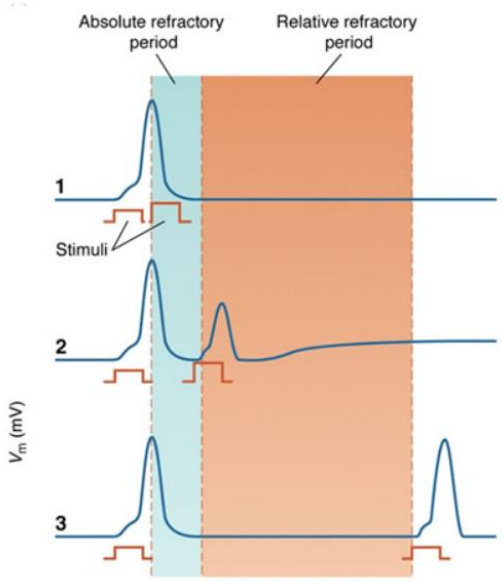
Η μεταβολή της τιμής του συνολικού δυναμικού της μεμβράνης σχετίζεται με τη διαδικασία ενεργοποίησης των νευρώνων, η οποία προκαλεί την παραγωγή χρονικά μεταβαλλόμενων ηλεκτρικών ιοντικών διαμεμβρανικών ρευμάτων, μέσω των επιλεκτικών καναλιών. Συνολικά υπάρχουν δύο κύριες μορφές νευρωνικής ενεργοποίησης, το δυναμικό δράσης (action potential) [32],[4] και το μετασυναπτικό δυναμικό (post-synaptic potential) [30],[32].

Το δυναμικό δράσης εμφανίζεται όταν το μεμβρανικό δυναμικό αντιστρέφεται αντιστρέψιμα ταχεία (5-10 ms [33]) από την τιμή ηρεμίας, περίπου $-70mV$, στην τιμή των $+50mV$ περίπου. Ο τρόπος διάδοσης του δυναμικού δράσης είναι κατά μήκος του νευρώνα και η μορφή του διακρίνεται στη συνέχεια.

Εικόνα 10 *Δυναμικό δράσης*

Η αρχή ενός δυναμικού δράσης είναι η αρχική αποπόλωση (depolarization) της μεμβράνης λόγω της παρουσίας ενός ερεθίσματος. Η φάση αυτή δεν οδηγεί απαραίτητα στην δημιουργία δυναμικού δράσης, καθώς για να συμβεί αυτό το ερέθισμα (δηλαδή η αποπόλωση του μεμβρανικού δυναμικού) πρέπει η αποπόλωση να είναι μεγαλύτερη από μία τιμή κατωφλίου $V_{threshold}$, η οποία σε κατάσταση ηρεμίας αντιστοιχεί περίπου στα $-50\,mV$. Η τιμή κατωφλίου αντιστοιχεί στην κατώτερη τιμή που είναι ικανή να παραγάγει δυναμικό δράσης καθώς είναι η τιμή του κατωφλίου αποκρισιμότητας των επιλεκττικών καναλιών ιόντων (κυρίως Na και K). Μετά την διέγερση ακολουθεί η επαναπόλωση (repolarizarion), κατά την οποία η τιμή του δυναμικού επανέρχεται στην τιμή που είχε σε κατάσταση ηρεμίας και περιλαμβάνει μια ενδιάμεση υπερπόλωσης (undershoot).

Τις χρονικές στιγμές που ο νευρώνας δεν είναι σε ηρεμία, η ουδός πυροδότησης δυναμικού δράσης αλλάζει εξαρτώμενη από τη φάση της διέγερσης στην οποία βρίσκεται. Αυτό μπορεί να καταστήσει αδύνατη την εκ νέου διέγερση. Το χρονικό διάστημα στο οποίο η τιμή του κατωφλίου διαφοροποιείται ονομάζεται περίοδος ανερεθιστότητας (refractory period). Πιο αναλυτικά, η περίοδος από την αρχή του δυναμικού δράσης μέχρι και το τέλος της κορυφής ονομάζεται περίοδος απόλυτης ανερεθιστότητας (absolute refractory period) και κατά την οποία είναι αδύνατο να προκληθεί νέα διέγερση, ανεξαρτήτως της έντασης του ερεθίσματος. Ακολουθεί μία περίοδος σχετικής ανερεθιστότητας (relative refractory period), στην οποία απαιτείται μεγαλύτερη ένταση ερεθίσματος από την ηρεμία για να πυροδοτηθεί νέο δυναμικό δράσης, είναι ωστόσο εφικτό.



Εικόνα 11 *Απεικόνιση της περιόδου ανερεθιστότητας*

Ο μέγιστος αριθμός δυναμικών δράσης που μπορούν να παραχθούν εντός ενός συγκεκριμένου χρονικού διαστήματος καθορίζεται από τη περίοδο απόλυτης ανερεθιστότητας ($\sim$1-2 ms) Με άλλα λόγια ισοδυναμεί με την ανώτερη συχνότητα απόκρισης των νευρώνων. Έτσι, αν η διάρκεια της απόλυτης ανερεθιστότητας είναι περίπου 1-2 ms, η μέγιστη συχνότητα απόκρισης κυμαίνεται μεταξύ 0.5-1kHz.

Αξίζει να σημειωθεί ότι εάν επιτευχθεί η τιμή κατωφλίου, ένα πλήρες δυναμικό δράσης παράγεται πάντα με συγκεκριμένο πλάτος $\sim$50mV που δεν μειώνεται καθώς διαδίδεται (λογική όλα-ή-τίποτα, all-or-nothing law). Στην πράξη αυτό μεταφράζεται στο ότι ανεξάρτητα με το πόσο ισχυρό είναι το ερέθισμα πάνω από την τιμή του κατωφλίου, θα παραχθεί ακριβώς ίδιο δυναμικό. Με άλλα λόγια, το δυναμικό δράσης εμφανίζεται μόνο όταν υπάρχει επαρκώς ισχυρό ερέθισμα και δεν αναπαριστά την ένταση του ερεθίσματος.. Η δύναμη ενός ερεθίσματος μεταφράζεται στην συχνότητα με την οποία παράγονται τα δυναμικά δράσης, δηλαδή ισχυρότερο ερέθισμα παράγει αναλογικά δυναμικά δράσης με μεγαλύτερη συχνότητα.

Από την άλλη μεριά, τα μετασυναπτικά δυναμικά είναι αργές αλλαγές στο δυναμικό της μεμβράνης που προκύπτουν από τη συναπτική ενεργοποίηση μέσω των νευροδιαβιβαστών. Αυτά τα δυναμικά χωρίζονται σε διεγερτικά (EPSPs - excitatory post-synaptic potentials) και κατασταλτικά (IPSPs - inhibitory post-synaptic potentials), με την επίδρασή τους να εξαρτάται από τον τύπο του νευροδιαβιβαστή, του υποδοχέα, καθώς και τις αλληλεπιδράσεις αυτών με κατάλληλα ιοντικά κανάλια (κυρίως Na+, K+ και Cl−). Πιο αναλυτικά, η αποπόλωση της μεμβράνης οδηγεί στα διεγερτικά δυναμικά (EPSPs), ενώ η υπερπόλωση της μεμβράνης οδηγεί στα κατασταλτικά (IPSPs). Αντίθετα με τα δυναμικά δράσης, τα οποία έχουν σταθερό πλάτος ανεξάρτητα από την ένταση του ερεθίσματος, τα μετασυναπτικά δυναμικά έχουν βαθμωτό πλάτος που εξαρτάται από την ένταση του ερεθίσματος και μειώνεται καθώς απομακρύνονται από το σημείο παραγωγής τους. Ακόμα, αυτά τα δυναμικά είναι ασυσχέτιστα με περίοδο ανερεθιστότητας και η διάρκειά τους μπορεί να φτάνει μέχρι και μερικά δευτερόλεπτα.

Το σήμα που καταγράφεται, λοιπόν, στην τεχνική της συμβατικής ηλεκτροεγκεφαλογραφίας προέρχεται από τα μετασυναπτικά δυναμικά και όχι από τα δυναμικά δράσης [3],[33]. Αν και τα μετασυναπτικά δυναμικά έχουν πολύ μικρότερο πλάτος σε σύγκριση με τα δυναμικά δράσης, τα χαρακτηριστικά τους, όπως η βαθμωτή τους φύση, η μεγαλύτερη διάρκεια και η ευρεία χωρική τους κατανομή, επιτρέπουν τη γεωμετρική άθροισή τους. Αυτό οδηγεί στη δημιουργία ενός μετρήσιμου

σήματος στην επιφάνεια του κεφαλιού, εφόσον τα μετασυναπτικά ρεύματα παρουσιάζουν χωρική και χρονική συσχέτιση. Αν δεν εξασφαλιστεί αυτό, τα επιμέρους σήματα αλληλοαναιρούνται και το γεωμετρικό άθροισμά τους δεν μπορεί να δώσει μετρήσιμο αποτέλεσμα, καθώς τα σήματα εξασθενούν λόγω της αντίστασης των στρωμάτων που μεσολαβούν μεταξύ των ηλεκτροδίων και των νευρώνων [4],[33].

Λαμβάνοντας υπόψη τα ανωτέρω είναι εύκολο να κατανοηθεί ότι η καταγεγραμμένη δραστηριότητα σε ένα μέρος της επιφάνειας του κεφαλιού περιέχει πλήθος διαφορετικών διεργασιών που συμβαίνουν την ίδια χρονική στιγμή σε διάφορα σημεία. Συνήθως, το πλάτος του καταγραφόμενου σήματος, το οποίο σχετίζεται με την έκταση της κατανομής του στην επιφάνεια του κεφαλιού, κυμαίνεται μεταξύ $0.5\mu V$ και $100\mu V$. Αυτό σημαίνει ότι είναι περίπου δύο τάξεις μεγέθους μικρότερο από τα τυπικά σήματα που καταγράφονται με ηλεκτροεγκεφαλογράφημα [34].

Ωστόσο υπάρχουν και τα δυναμικά που σχετίζονται με γεγονότα (Event-Related Potentials - ERPs) και είναι διαφορές δυναμικού που καταγράφονται όταν ο εξεταζόμενος εκτίθεται σε συγκεκριμένα ερεθίσματα κατά τη διάρκεια δοκιμασιών. Αυτά τα δυναμικά σχετίζονται με το αντίστοιχο ερέθισμα ή γεγονός, είτε εσωτερικό είτε εξωτερικό, τόσο σε επίπεδο χρόνου (time-locked), όσο και σε επίπεδο φάσης (phase-locked). Η χρονική συσχέτιση σημαίνει ότι η απόκριση εμφανίζεται σε γενικές γραμμές με την ίδια μορφή σε δεδομένο χρόνο μετά την παρουσία του ερεθίσματος, ενώ η φασική συσχέτιση είναι πιο «αυστηρή» και υποδεικνύει ότι η απόκριση έχει την ίδια φάση σε σχέση με το ερέθισμα κάθε φορά. Αυτή η δραστηριότητα, που είναι συγχρονισμένη τόσο στον χρόνο όσο και στη φάση, αναφέρεται ως «προκλητή» (evoked) και τα δυναμικά που σχετίζονται με γεγονότα είναι ευρέως γνωστά ως προκλητά δυναμικά (evoked potentials) ή ΠΔ. [3],[4],[33]

Για να αξιοποιηθούν σωστά τα προκλητά δυναμικά στην έρευνα της εγκεφαλικής και νοητικής λειτουργίας, είναι σημαντικό να ληφθούν υπόψη παράγοντες όπως η εξοικείωση του ατόμου με το ερέθισμα, η κόπωση, καθώς και οι εξωτερικές και εσωτερικές συνθήκες. Εξωτερικοί παράγοντες, όπως το περιβάλλον, και εσωτερικοί, όπως η ψυχολογική κατάσταση του εξεταζόμενου, μπορούν να επηρεάσουν τα αποτελέσματα.

Η παρουσία ενός εξωτερικού ερεθίσματος δεν προκαλεί μόνο παραγωγή προκλητών δυναμικών, όμως, αλλά οδηγεί και στην ύπαρξη σχετικού με το γεγονός συγχρονισμού (event-relates synchronization - ERS) ή αποσυγχρονισμού (event-related desynchronisation - ERD) [35]. Η κύρια διαφορά μεταξύ αυτών και των προκλητών δυναμικών, είναι ότι ενώ τα προκλητά δυναμικά είναι time-locked και phase-locked, ο συγχρονισμός και αποσυγχρονισμός είναι μόνο time-locked [36],[37]. Αυτή η δραστηριότητα ονομάζεται επαγόμενη (induced). Ο συγχρονισμός και ο αποσυγχρονισμός αναφέρονται στη συμπεριφορά πληθυσμών νευρώνων για κάποια συχνότητα και κατά πόσο αυτή είναι αυξημένα ή μειωμένα συγχρονισμένη και, μάλιστα, μπορεί να εμφανιστεί ταυτόχρονα ERD και ERS για διαφορετικές συχνότητες σε διαφορετικές περιοχές του εγκεφάλου [35]. Μαθηματικά, η ύπαρξη συγχρονισμού/αποσυγχρονισμού σε κάποια συχνότητα ορίζεται μέσω της ύπαρξης θετικής, για ERS, ή αρνητικής, για ERD, κορυφής στην ισχύ του σήματος.

6.3. Βασικοί Ρυθμοί Εγκεφαλογραφήματος

Το ηλεκτροεγκεφαλογραφικό σήμα διαχωρίζεται σε ρυθμούς ανάλογα με το εύρος της βασικής συχνότητας των κυμάτων του. Υπάρχουν πέντε κύριοι ρυθμοί, των οποίων οι ζώνες συχνοτήτων επικαλύπτονται μεταξύ τους, με τον διαχωρισμό να είναι κυρίως προσεγγιστικός [29],[4],[33]. Στη συνέχεια, παρουσιάζονται περιληπτικά με κάποια βασικά στοιχεία τους. Άξια προσοχής είναι η αρνητική συσχέτιση μεταξύ πλάτους και συχνότητας των κυμάτων [35].

6.3.1. Ρυθμός α (alpha rhythm)

Εικόνα 12 *Ρυθμός α*

Ιστορικά ο ρυθμός α αποκαλείται και κύματα Berger προς τιμήν του Hans Berger, ο οποίος ήταν ο πρώτος που τα περιέγραψε όταν εφηύρε το ηλεκτροεγκεφαλογράφημα το 1924. Είναι ο πρώτος εγκεφαλικός ρυθμός που καταγράφηκε στον άνθρωπο [38].

Το πιο συχνά αναφερόμενο εύρος του είναι στα 8-13Hz [33], αν και μπορεί να παρατηρηθεί και σχετική κορυφή σε συχνότητες που ανήκουν σε άλλες ζώνες όπως στα 20Hz και τα 75Hz. Το πλάτος του συνήθως δεν ξεπερνά τα $50\mu V_{pp}$. Εμφανίζεται κυρίως στις οπίσθιες και ινιακές περιοχές του εγκεφάλου, ενώ μπορεί να καταγραφεί και στον μετωπιαίο λοβό κατά την κατάσταση χαλάρωσης. [30],[4]

Ο ρυθμός α μπορεί περαιτέρω να χωριστεί σε δύο υποκατηγορίες τους ρυθμούς μ και τ:

- Ρυθμός μ (mu rhythm): Αντιστοιχεί στην κεντρική περιοχή της ζώνης α και αφορά τον μυικό και τον σωματοαισθητικό φλοιό. [29],[4],[39]
- Ρυθμός τ (tau rhythm): Εμφανίζεται περίπου στα 10Hz και σχετίζεται με τον κροταφικό λοβό. Γενικά, εξασθενεί λόγω ακουστικών ερεθισμάτων. [4],[39]

Η ύπαρξη έντονης δραστηριότητας α στις μετωπιαίες περιοχές συνήθως σχετίζεται με κατάσταση χαλάρωσης, ενώ εμφανίζεται επίσης με το κλείσιμο των ματιών στις οπίσθιες και ινιακές περιοχές [30],[4]. Κανονικά η δραστηριότητα αυτή περιορίζεται με το άνοιγμα των ματιών, την εγρήγορση, τη σκέψη και την πραγματοποίηση οποιασδήποτε γνωσιακής εργασίας. Έχει παρατηρηθεί ότι ο φωτισμός υποβάθρου (background illumination) μπορεί να μειώσει το πλάτος του [29].

6.3.2. Ρυθμός β (beta rhythm)

Εικόνα 13 *Ρυθμός β*

Το εύρος συχνοτήτων του είναι περίπου 14-26Hz, ωστόσο αναφέρεται συχνά ότι εμπεριέχει τις συχνότητες μέχρι και τα 30Hz και μπορεί να χωριστεί περαιτέρω στα Χαμηλά Κύματα Βήτα (14–16 Hz), τα Κύματα Βήτα (16.5–20 Hz) και τα Υψηλά Κύματα Βήτα (20.5-26 Hz). Το πλάτος τους δεν υπερβαίνει τα $30\mu V$ και εμφανίζεται κυρίως σε μετωπιαίες και κεντρικές περιοχές [40].

Είναι ο ρυθμός που επικρατεί όταν το άτομο είναι σε κατάσταση εγρήγορσης και έχει τα μάτια ανοιχτά [4],[33]. Σχετίζεται λοιπόν με την αυξημένη εγρήγορση, την προσοχή και την επίλυση προβλημάτων [39]. Επίσης, ο ρυθμός β μπορεί να εμφανιστεί σε καταστάσεις πανικού, ενώ η δραστηριότητα στον κινητικό φλοιό σχετίζεται με την προετοιμασία για κίνηση είτε πραγματική είτε φανταστική [41].

6.3.3. Ρυθμός γ (gamma rhythm)

Εικόνα 14 *Ρυθμός γ*

Ως ρυθμός γ χαρακτηρίζονται όλες οι συχνότητες άνω των 30Hz. Το πλάτος τους είναι χαμηλό και καταγράφεται κυρίως στην κεντρομετωπιαία (frontocentral) περιοχή του εγκεφάλου. [4]

Καταγράφεται σχετικά σπάνια [4] και σχετίζεται με την αντίληψη και την γρήγορη επεξεργασία πληροφοριών [35]. Επιπλέον, εμφανίζεται κατά την εναλλαγή μεταξύ διαφορετικών εργασιών και στην παράλληλη επεξεργασία πληροφοριών, όπου εμπλέκεται σημαντικά η βραχυπρόθεσμη μνήμη.

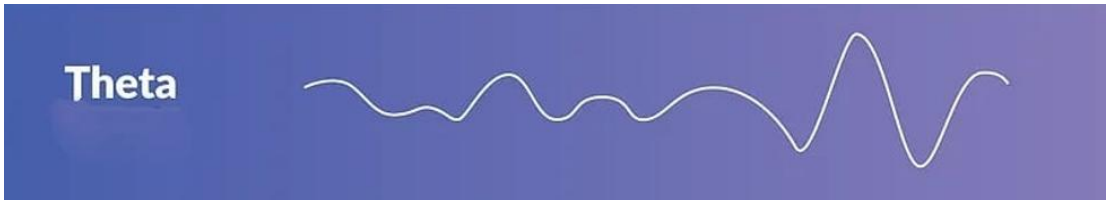
6.3.4. Ρυθμός δ (delta rhythm)

Εικόνα 15 *Ρυθμός δ*

Τα κύματα δ περιγράφτηκαν αρχικά από τον Willian Grey Walter την δεκαετία του 1930, όταν βελτίωσε το ηλεκτροεγκεφαλογράφημα του Berger και κατέστη δυνατή η καταγραφή τους [42]. Περιλαμβάνει περίπου το εύρος 0.5-4 Hz, δηλαδή πρακτικά όλες τις συχνότητες κάτω της ζώνης α και το πλάτος του μπορεί να φτάσει μέχρι τις μερικές εκατοντάδες μV [4],[33].

Εμφανίζεται κυρίως σε κατάσταση χαλάρωσης ή ύπνου και καταστέλλεται σε καταστάσεις αναταραχής ή εγρήγορσης. Αυτός ο ρυθμός αποτελεί τον βασικό ρυθμό έως το δεύτερο έτος της ηλικίας του ανθρώπου. Επιπλέον, μελέτες έχουν δείξει αύξησή του ρυθμού δ κατά την εκτέλεση νοητικών διεργασιών [43], [44], [45],[46].

6.3.5. Ρυθμός θ (theta rhythm)

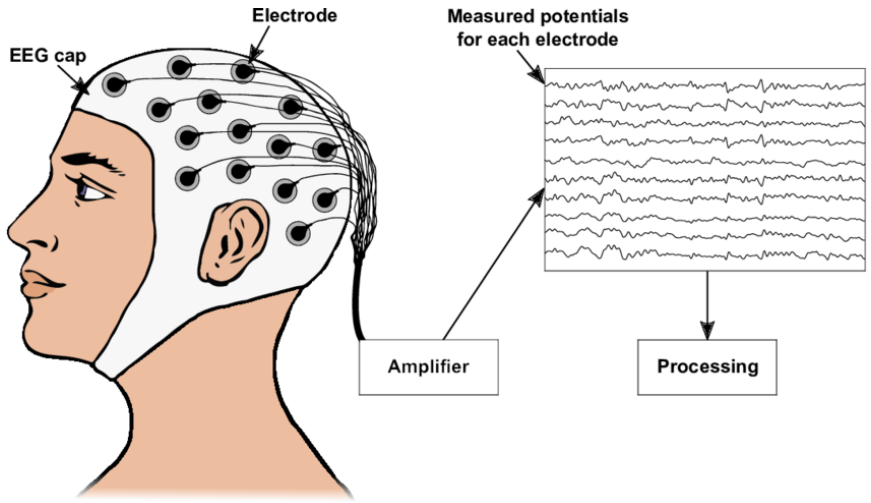


Εικόνα 16 *Ρυθμός θ*

Ο ρυθμός θ θεωρείται ότι προέρχεται από τον θάλαμο. Η πρώτη αναφορά για κανονικές αργές ταλαντώσεις του ιππόκαμπου έγινε το 1938 χωρίς ωστόσο τη δυνατότητα περαιτέρω διερεύνησης από τους αρχικούς παρατηρητές Jung και Kornmüller. Το 1954 έγιναν διαθέσιμες περισσότερες πληροφορίες χάρη στην λεπτομερή έρευνα των John D. Green και Arnaldo Arduini. Ο ρυθμός θ περιλαμβάνει το εύρος συχνοτήτων 4-8 Hz και σε πλάτος μπορεί να φτάσει τα 100μV [47]. Λόγω της προέλευσής του καταγράφεται κυρίως από τη μέση μετωπιαία (frontal midline) περιοχή [48].

Σχετίζεται με τη δημιουργικότητα, τη διαίσθηση, την ονειροπόληση και τη φαντασίωση. Τα κύματα θ είναι ισχυρά σε καταστάσεις όπως είναι η χαλάρωση, ο διαλογισμός και η ανάκληση [4].

6.4. Λήψη Σήματος



Εικόνα 17 *Σύστημα καταγραφής ηλεκτοεγκεφαλογραφήματος*

Το σύστημα καταγραφής ενός ηλεκτροεγκεφαλογραφήματος περιλαμβάνει τα εξής [30],[33]:

- Ηλεκτρόδια καταγραφής
- Διατάξεις ενίσχυσης και φιλτραρίσματος
- Διάταξη μετατροπής αναλογικοψηφιακού σήματος (A/D converter)
- Σύστημα για καταγραφή, αποθήκευση και προβολή μετρήσεων
- Προαιρετικά σύστημα χορήγησης ερεθισμάτων

6.4.1. Διαδικασία

Η καταγραφή του σήματος για το ηλεκτροεγκεφαλογράφημα στηρίζεται στη διαφορική ενίσχυση (differential amplification), του σήματος μεταξύ δύο ηλεκτροδίων. Κάθε ενισχυτής διαφορικής ενίσχυσης αντιστοιχεί σε ένα κανάλι καταγραφής [34] και έχει δύο εισόδους (αναστρέφουσα και μη αναστρέφουσα) για ηλεκτρόδια, που συνθέτουν μία απαγωγή. Το σύνολο των απαγωγών αποτελεί ένα μοντάζ. Υπάρχουν πολλά είδη μοντάζ, τα οποία περιγράφονται παρακάτω, τα οποία μπορούν να επιδράσουν στον τύπο της δραστηριότητας στον οποίο θα δοθεί έμφαση μέσω της διαφορικής ενίσχυσης. Πιο αναλυτικά, κάποια είδη μοντάζ εστιάζουν στην ανάδειξη δραστηριότητας εντοπισμένης σε μία ή πολλές περιοχές, ενώ άλλα είδη αναδεικνύουν καλύτερα δραστηριότητα που είναι ευρέως κατανεμημένη στην επιφάνεια του κεφαλιού. Αν όλα τα ηλεκτρόδια που είναι διαθέσιμα συσχετιστούν με οποιονδήποτε τρόπο μέσω της επιλογής των αναφορών, η ανακατασκευή οποιουδήποτε μοντάζ είναι δυνατή. Αξίζει, τέλος, να σημειωθεί ότι κατά τη διάρκεια της καταγραφής μπορεί να χρησιμοποιηθεί ένα επιπλέον ηλεκτρόδιο γείωσης για τον ορισμό αναφοράς της μηδενικής τάσης του σήματος [30],[33].

Τα κυριότερα μοντάζ παρατίθενται στη συνέχεια [34]:

- Διπολικό (Bipolar): Κάθε απαγωγή αντιστοιχεί σε ένα ζεύγος γειτονικών ηλεκτροδίων, τα οποία μπορούν να ορίζονται είτε κατά τον οριζόντιο άξονα (πρόσθιο - οπίσθιο), είτε κατά τον εγκάρσιο (αριστερό - δεξιό). Τα ζεύγη αυτά είναι διαρθρωμένα σε μορφή αλυσίδας, όπου κάθε

ηλεκτρόδιο που συνδέεται στην αναστρέφουσα είσοδο ενός ενισχυτή είναι επίσης συνδεδεμένο με τη μη αναστρέφουσα είσοδο του επόμενου ενισχυτή. Αυτή η διασύνδεση καθιστά τα διπολικά μοντάζ πιο κατάλληλα για την ανάλυση τοπικής δραστηριότητας, ενώ παράλληλα απορρίπτουν σήματα που κατανέμονται σε ευρύτερες περιοχές.

- Κοινού ηλεκτροδίου αναφοράς (Common electrode reference): Σε αυτή την τεχνική, ένα συγκεκριμένο ηλεκτρόδιο χρησιμοποιείται ως κοινή βάση αναφοράς για όλους τους ενισχυτές. Αυτή η προσέγγιση είναι ιδιαίτερα αποτελεσματική για την ανίχνευση ευρέως κατανεμημένης δραστηριότητας, ενώ τα σήματα που καταγράφονται έχουν μεγαλύτερο πλάτος, εξαιτίας των αυξημένων αποστάσεων μεταξύ των ηλεκτροδίων, συγκριτικά με τη μέθοδο του διπολικού μοντάζ. Η θέση του ηλεκτροδίου αναφοράς μπορεί να επιλέγεται τόσο από την περιοχή της κεφαλής όσο και από περιοχές εκτός αυτής. Ιδανικά, το ηλεκτρόδιο τοποθετείται σε σημείο όπου επικρατεί ηρεμία και παρατηρείται ελάχιστη ηλεκτρική δραστηριότητα, όπως οι μαστοειδείς αποφύσεις ή οι λοβοί των αυτιών [30].
- Μέσης αναφοράς (Average reference): Σε αυτή τη μέθοδο, το σήμα κάθε ηλεκτροδίου συγκρίνεται με το συνολικό άθροισμα των σημάτων όλων των ηλεκτροδίων. Με αυτόν τον τρόπο, το σήμα κάθε ηλεκτροδίου αντιπαραβάλλεται τόσο με το συλλογικό σήμα των υπολοίπων όσο και με το δικό του σήμα. Αυτή η προσέγγιση είναι ιδιαίτερα κατάλληλη για την ανίχνευση και την ανάλυση δραστηριότητας που εκτείνεται σε ευρείες περιοχές.
- Αναφοράς σταθμισμένου μέσου (Weighted average reference): Αυτό το μοντάζ αποτελεί μια τροποποιημένη εκδοχή του μοντάζ μέσης αναφοράς, με δύο κύριες διαφοροποιήσεις. Πρώτον, τα ηλεκτρόδια δεν συμβάλλουν ισότιμα στο σήμα αναφοράς. Αντίθετα, τα ηλεκτρόδια που βρίσκονται πιο κοντά στο υπό εξέταση ηλεκτρόδιο έχουν μεγαλύτερη συνεισφορά. Οι συντελεστές βάρους για τη συμβολή κάθε ηλεκτροδίου μπορούν να προσδιοριστούν μαθηματικά, λαμβάνοντας υπόψη τις πραγματικές αποστάσεις από το κεντρικό ηλεκτρόδιο, συνήθως με τρόπο αντιστρόφως ανάλογο της απόστασης. Η δεύτερη διαφορά είναι ότι το σήμα του βασικού ηλεκτροδίου, δηλαδή του ηλεκτροδίου που συνδέεται με τη δεύτερη είσοδο του διαφορικού ενισχυτή, δεν περιλαμβάνεται στον υπολογισμό της αναφοράς για το συγκεκριμένο κανάλι. Αυτό το μοντάζ είναι πιο κατάλληλο για την ανάδειξη εντοπισμένης δραστηριότητας, σε σύγκριση με την απλή μέση αναφορά.
- Μέσης αναφοράς προέλευσης (Laplacian – source derivation): Στο μοντάζ αυτό στο σήμα αναφοράς συνεισφέρουν μόνο τα γειτονικά ηλεκτρόδια. Για αυτόν τον λόγο βασικό μειονέκτημα αυτής της μεθόδου είναι το περιορισμένο εύρος ηλεκτροδίων που μπορούν να χρησιμοποιηθούν στα όρια των περιοχών κάλυψης του κεφαλιού.

Αναφορικά με την ενισχυτική διάταξη [34], αυτή πρέπει να πληροί κάποιες βασικές προϋποθέσεις που υπάρχουν για κάθε βιοενισχυτή [49]. Αρχικά, ο ενισχυτής δεν πρέπει να επηρεάζει τη διαδικασία που καταγράφεται, να παραμορφώνει ελάχιστα (ιδανικά καθόλου) το σήμα, και να διαχωρίζει αποτελεσματικά το επιθυμητό σήμα από τον θόρυβο. Επιπλέον, είναι σημαντικό να προστατεύει πλήρως τον εξεταζόμενο από κινδύνους λόγω του ηλεκτρικού ρεύματος ή τοξικών υλικών αλλά και να προστατεύει την ίδια την διάταξη.

Ειδικότερα, όταν πρόκειται για διάταξη καταγραφής επιφανειακού ηλεκτροεγκεφαλογραφήματος, το πολύ μικρό πλάτος των παραγόμενων σημάτων, το οποίο συνήθως κυμαίνεται από μερικές δεκάδες nV μέχρι λίγες εκατοντάδες μV [33], απαιτεί υψηλό κέρδος ενίσχυσης έως και της τάξεως του 10^4 [49] και πολύ χαμηλού θορύβου [33], με λόγο απόρριψης κοινού σήματος τουλάχιστον 100dB [49]. Ο θόρυβος μιας διάταξης μπορεί ευκόλως να βρεθεί βραχυκυκλώνοντας τις εισόδους της και μετρώντας την τάση εξόδου. Τέλος η αντίσταση εισόδου πρέπει να είναι τουλάχιστον 100MΩ [30].

Σχετικά με το πεδίο συχνότητας, οι διατάξεις γενικά σχεδιάζονται με σκοπό να αποκόπτουν συχνότητες κάτω από 0.5Hz και πάνω από 70Hz, διότι εντός αυτών των ορίων είναι όλες οι κλινικώς χρήσιμες συχνότητες [4]. Πιο αναλυτικά, απαιτείται η χρήση υψιπερατού φίλτρου με συχνότητα αποκοπής γύρω στα 0.1-0.7Hz για την εξάλειψη συχνοτήτων που παράγονται από διεργασίες όπως η αναπνοή, και η χρήση βαθυπερατού φίλτρου με συχνότητα αποκοπής από 40Hz μέχρι και το μισό του ρυθμού δειγματοληψίας, έτσι ώστε να μην εμφανιστούν φαινόμενα αναδίπλωσης [33]. Τέλος, για την απόρριψη των παρεμβολών λόγω συχνότητας τροφοδοσίας μπορεί να χρησιμοποιηθεί ένα στοχευμένο ζωνοφρακτικό φίλτρο [4]. Σε επόμενο στάδιο, μετά την ψηφιοποίηση του σήματος μπορεί να γίνει χρήση ψηφιακών φίλτρων [33]. Για την ψηφιοποίηση του σήματος συνιστάται χρήση μετατροπέα A/D τουλάχιστον 12bits με ανάλυση το λιγότερο 0.5μV [30]. Αυτονόητο θεωρείται το γεγονός ότι η συχνότητα δειγματοληψίας πρέπει να ικανοποιεί το κριτήριο Nyquist, δηλαδή να είναι τουλάχιστον διπλάσια της μέγιστης συχνότητας που αναλύεται. Παραδείγματος χάριν, αν θεωρηθεί ότι το χρήσιμο εύρος ζώνης φτάνει μέχρι περίπου τα 100Hz [50], τότε η ελάχιστη συχνότητα δειγματοληψίας που απαιτείται θα είναι 200Hz.

6.4.2. Εξοπλισμός

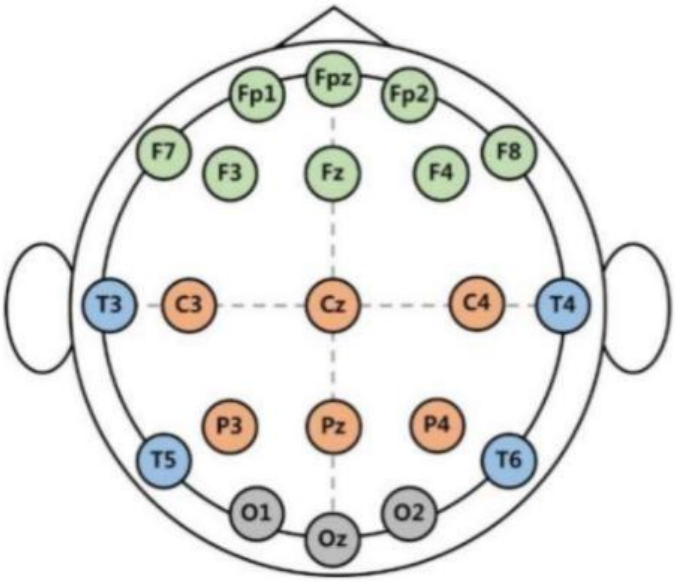
Για τη λήψη του ηλεκτροεγκεφαλογραφήματος χρησιμοποιούνται ηλεκτρόδια τα οποία τοποθετούνται στο κεφάλι. Υπάρχουν διάφοροι τύποι ηλεκτροδίων, τα τυπικά δισκοειδή , τα ηλεκτρόδια αλατούχου διαλύματος και τα ηλεκτρόδια βελόνας. Εκτός από μεμονωμένα ηλεκτρόδια συνήθως χρησιμοποιούνται οι κάσκες ηλεκτροδίων, ιδιαίτερα όταν απαιτείται μεγάλος αριθμός ηλεκτροδίων για την διεκπεραίωση μίας μέτρησης. Για τη λήψη του συμβατικού επιφανειακού ηλεκτροεγκεφαλογραφήματος, συνήθως χρησιμοποιούνται ηλεκτρόδια αργύρου/χλωριούχου αργύρου (Ag/AgCl) με διάμετρο 1-3mm [30],[4],[33]. Η αντίσταση επαφής μεταξύ ηλεκτροδίου και δέρματος πρέπει να είναι μικρότερη από 5kΩ και η διαφορά της μεταξύ των ηλεκτροδίων δεν πρέπει να ξεπερνά το 1kΩ [4]. Για τον λόγο αυτό, είναι συνηθισμένο να καθαρίζεται το δέρμα και να εφαρμόζεται ηλεκτρολυτική αλοιφή [30]. Συνήθως οι διατάξεις περιλαμβάνουν σύστημα ελέγχου της αντίστασης της επαφής ηλεκτροδίου-δέρματος, το οποίο μετρά την αντίσταση κάθε ζεύγους ηλεκτροδίων μέσω της διοχέτευσης μικρής ποσότητας ρεύματος [33].

Η τοποθέτηση των ηλεκτροδίων στοχεύει στην κάλυψη της μέγιστης δυνατής επιφάνειας του κεφαλιού [33]. Παρά το γεγονός ότι η αύξηση του αριθμού των ηλεκτροδίων βελτιώνει τη χωρική ανάλυση [34], εάν δύο ηλεκτρόδια είναι πολύ κοντά, η δραστηριότητα που καταγράφουν γίνεται όλο και πιο όμοια. Αντίθετα, η αύξηση της απόστασης μεταξύ δύο ηλεκτροδίων ενισχύει τη διαφορά του σήματος που καταγράφουν, κάτι που γίνεται εμφανές μέσω διαφορικής ενίσχυσης. Παρ' όλα αυτά, όταν η απόσταση υπερβαίνει τα 10 cm, η επιπλέον αύξηση δεν προσφέρει σημαντική βελτίωση στο πλάτος του μετρούμενου σήματος [34]. Άλλωστε, αν η απόσταση μεταξύ των σημείων καταγραφής είναι μεγαλύτερη από την έκταση της δραστηριότητας που ανιχνεύεται, μπορεί να προκληθεί υποδειγματοληψία (undersampling), κάτι που οδηγεί σε χωρική αναδίπλωση (spatial aliasing).

Όσον αφορά τις θέσεις τοποθέτησης των ηλεκτροδίων, στο πέρασμα των χρόνων έχει προταθεί πλήθος προτύπων, με τα πιο γνωστά να είναι το σύστημα 10/20 [30],[33], το οποίο είναι και το συνηθέστερο, το 10/10 ή 10% και το 10/5 ή 5%, τα οποία έχουν πιο πυκνή τοποθέτηση [51],[52]. Πιθανό είναι επίσης να εφαρμοστούν διάφορες παραλλαγές των συστημάτων αυτών με την προσθήκη επιπλέον ηλεκτροδίων.

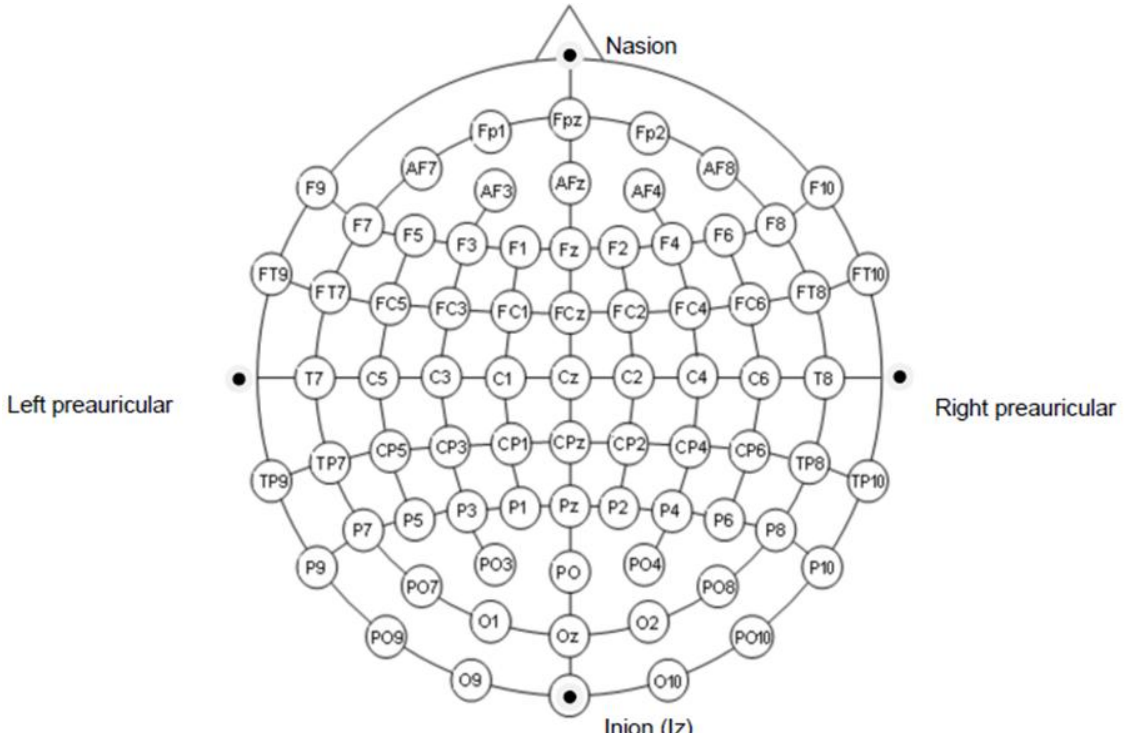
Σύμφωνα με το σύστημα 10/20, η επιφάνεια του κεφαλιού διαιρείται βάσει των αποστάσεων μεταξύ των τεσσάρων σημείων αναφοράς, τα οποία είναι το ριζορρίνιο (nasion), το ινίο (inion) και τα δύο προωτιαία σημεία (preauricular points). Κάθε ηλεκτρόδιο έχει ένα μοναδικό όνομα ανάλογα με τη θέση στην οποία τοποθετείται. Το όνομα αποτελείται από ένα ή δυο γράμματα και ακολουθείται από

έναν αριθμό ή το γράμμα z. Το αρχικό γράμμα υπάρχει για να χαρακτηρίζει τον λοβό ή την περιοχή του κεφαλιού από την οποία το ηλεκτρόδιο καταγράφει, συγκεκριμένα είναι προμετωπιαίος (pre-frontal - Fp), μετωπιαίος (frontal - F), κροταφικός (temporal - T), βρεγματικός (pariental - P), ινιακός (occipital - O) και κεντρική θέση (central - C). Όταν το όνομα ενός ηλεκτροδίου καταλήγει στο λατινικό γράμμα z, σημαίνει ότι το ηλεκτρόδιο είναι τοποθετημένο στην γραμμή που ορίζουν το ριζορρίνιο και το ινίο. Αντιθέτως, στην γραμμή μεταξύ των δύο προωτιαίων σημείων τοποθετούνται τα T3, C3, Cz (που είναι πάνω στην τομή των δύο αξόνων), C4 και T4. Επί του κύκλου με κέντρο το Cz και ακτίνα το T3 τοποθετούνται τα F7, Fp1, Fp2, F8, T6, O2, O1 και T5. Το F3 τοποθετείται έτσι ώστε να ισαπέχει από τα F7, Fz και να βρίσκεται επί του κύκλου με κέντρο το T4 και ακτίνα ίση με την απόσταση μεταξύ T4 και C3. Τέλος, τα F4, P3, P4 τοποθετούνται συμμετρικά του F3 ως προς τους δύο άξονες που ορίζονται από τα σημεία αναφοράς. Γενικά, οι περιττοί (άρτιοι) αριθμοί αντιστοιχούν στο αριστερό (δεξί) ημισφαίριο.



Εικόνα 18 *Σύστημα 10/20*

Οι παραπάνω ονομασίες δεν μεταφέρονται αυτούσιες στα συστήματα των οποίων η χρήση απαιτεί μεγαλύτερο αριθμό ηλεκτροδίων. Για παράδειγμα, τα ηλεκτρόδια T3, T4, T5 και T6 μετονομάζονται σε T7, T8, P7 και P8 αντίστοιχα [53]. Επιπλέον, στο σύστημα 10/10 έχουν εισαχθεί νέες ονομασίες για τον χαρακτηρισμό των επιπρόσθετων θέσεων που εισάγονται ανάμεσα στις θέσεις του συστήματος 10/20 [53]. Οι θέσεις αυτές είναι οι εμπροσθομετωπιαίες (anterior frontal – AF), οι κεντρομετωπιαίες (frontocentral – FC), οι μετωποκροταφικές (frontotemporal – FT), οι οπισθοκροταφικές (temporal posterior – TP), οι κεντροβρεγματικές (centroparietal – CP) και οι βρεγματοϊνιακές (parieto-occipital – PO).



Εικόνα 19 *Σύστημα 10/10*

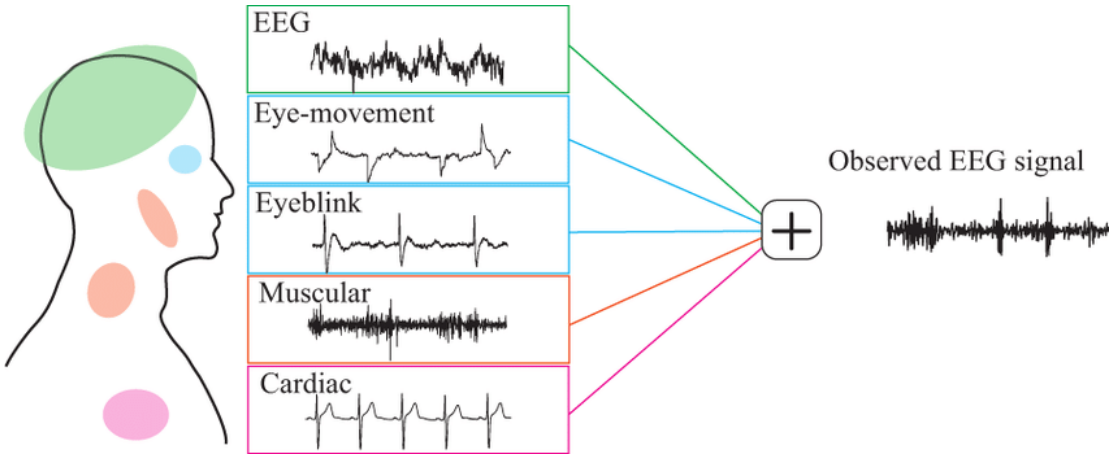
6.5. Επεξεργασία Σήματος

6.5.1. Προεπεξεργασία Δεδομένων

Η προεπεξεργασία είναι η διαδικασία της μετατροπής ενός ακατέργαστου (raw) σήματος σε μορφή η οποία είναι κατάλληλη για περαιτέρω ανάλυση και ερμηνεία από τον χρήστη. Στην περίπτωση του ηλεκτροεγκεφαλογραφήματος, η προεπεξεργασία συνήθως αναφέρεται στην αφαίρεση του θορύβου από τα δεδομένα με στόχο την προσέγγιση των πραγματικών νευρικών σημάτων. Το σήμα εμφανίζει συχνά παραμορφώσεις, οι λόγοι των οποίων μπορούν να ταξινομηθούν σε δύο κατηγορίες, τις φυσιολογικές και τις τεχνικές [34],[54],[55].

Οι φυσιολογικές αιτίες ονομάζονται επιπλέον εσωτερικές ή βιολογικές και οφείλονται σε ηλεκτρικά δυναμικά και κινήσεις του οργανισμού. Για παράδειγμα, η κίνηση του σώματος και ειδικά η κίνηση της κεφαλής αποτυπώνεται ως δραστηριότητα στη ζώνη δ. Η μυϊκή κίνηση αποτυπώνεται κυρίως ως σύντομη χρονικά κορυφή στη ζώνη β, ενώ η εμφάνιση της και στη ζώνη α είναι πιθανή. Παράδειγμα με ιδιαίτερο ενδιαφέρον αυτής της κατηγορίας είναι η κίνηση του μετώπου [54],[55], που συνδυάζει χαρακτηριστικά τόσο σωματικής όσο και μυϊκής δραστηριότητας και εμφανίζεται κυρίως σε συχνότητες άνω των 13 Hz και δευτερευόντως στη ζώνη δ. Ένα άλλο παράδειγμα της ίδιας κατηγορίας είναι το σφίξιμο του σαγονιού [54],[55] που εμφανίζεται στις άνω των 13Hz συχνότητες. Η κίνηση των ηλεκτροδίων που προκαλούνται από τους παλμούς της καρδιάς αποτυπώνεται και αυτή στη ζώνη δ. Οι οφθαλμικές ατέλειες, δηλαδή τόσο το κλείσιμο των οφθαλμών όσο και οι κινήσεις αυτών, αποτελούν επίσης φυσιολογικές αιτίες εμφάνισης ατελειών (artifacts). Και οι δύο αυτές ατέλειες εμφανίζονται στις χαμηλές δ συχνότητες και επηρεάζουν κυρίως τα μετωπιαία ηλεκτρόδια. Πιο συγκεκριμένα, το άνοιγμα και το κλείσιμο εμφανίζουν δραστηριότητα μίας τάξης μεγέθους μεγαλύτερη [4] του επιθυμητού σήματος στις θέσεις Fp1 και Fp2, ενώ οι οφθαλμικές κινήσεις αλλοιώνουν τις θέσεις πάλι των Fp1 και Fp2 για κατακόρυφες κινήσεις και F7 και F8 για οριζόντιες [34], δημιουργώντας έτσι ασυμμετρία μεταξύ των ημισφαιρίων [54],[55]. Ακόμα, η γλωσσοκινητική ατέλεια, η οποία δημιουργείται λόγω της κίνησης της γλώσσας, κυρίως εμφανίζεται στη μετωπιαία περιοχή με δραστηριότητα στη ζώνη δ. Στη ζώνη δ εμφανίζεται αι η αναπνοή ως ρυθμική

δραστηριότητα. Τέλος, και ο ιδρώτας αποτελεί σημαντικό βιολογικό αίτιο ατελειών καθώς εμπλέκεται στην επαφή μεταξύ δέρματος και ηλεκτροδίου.



Εικόνα 20 *Διαχωρισμός σήματος ΗΕΓ σε σήματα αναλόγως με την πηγή τους*

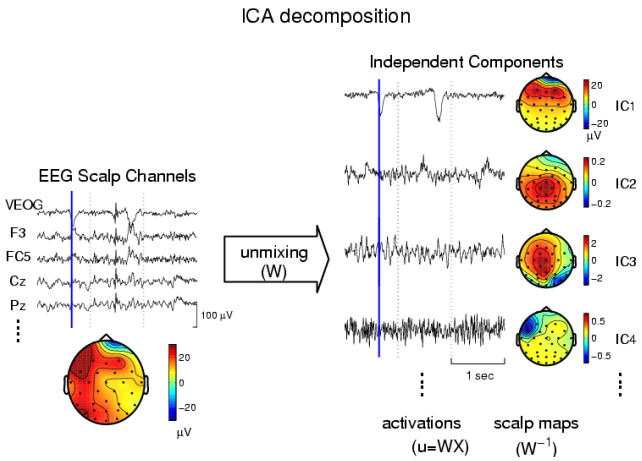
Από την άλλη μεριά, οι τεχνικές αιτίες (ατέλειες συστήματος ή μη-φυσιολογικές) εμφανίζονται λόγω ηλεκτρικών πηγών πέραν του σώματος του εξεταζόμενου. Αυτές μπορεί να περιλαμβάνουν την παρεμβολή που οφείλεται στην ηλεκτρική τροφοδοσία, τις αυξομειώσεις της αντίστασης της επαφής μεταξύ δέρματος και ηλεκτροδίου, τις μεγάλες διαφορές μεταξύ των αντιστάσεων για διαφορετικά ηλεκτρόδια, τα προβλήματα/κινήσεις των καλωδίων, τον θόρυβο από τα ηλεκτρονικά εξαρτήματα, και την ανεπαρκή τροφοδοσία.

Η συχνότητα κάθε κυματομορφής που εμφανίζεται στο σήμα και μπορεί να ερμηνευτεί ως ατέλεια έχει ιδιαίτερη σημασία, καθώς έχει άμεση συσχέτιση με το αίτιο εμφάνισής της. Παράδειγμα είναι οι ατέλειες στη ζώνη δ και ιδιαίτερα περίπου στο εύρος 0.8-1.7Hz (60-100bpm [56]), το οποίο είναι το τυπικό εύρος συχνότητας των καρδιακών παλμών. Επομένως, η συχνότητα παίζει πολύ σημαντικό ρόλο στην εξαγωγή της χρήσιμης πληροφορίας από το αρχικό σήμα. Για αυτόν το λόγο ένα από τα βασικότερα στάδια επεξεργασίας είναι η ανάλυση στο πεδίο της συχνότητας με χρήση του Μετασχηματισμού Fourier (FFT), μέσω του οποίου μπορεί να ληφθεί το φάσμα ισχύος (power spectrum) και να γίνει μελέτη των διαφορετικών ζωνών συχνοτήτων. Γενικά, ο FFT ενδείκνυται για ανάλυση στάσιμων (stationary) σημάτων και όχι μη στάσιμων (non stationary), κατηγορία στην οποία ανήκει και το ηλεκτροεγκεφαλογράφημα. Παρ' όλα αυτά, χρησιμοποιείται ευρέως για αυτόν τον σκοπό, χάρη στη μεγάλη του ταχύτητα για εφαρμογές σε πραγματικό χρόνο και στην ευκολία που παρέχει στην ανάλυση ζωνών συχνοτήτων. Η εναλλακτική, καταλληλότερη, τεχνική για ανάλυση σημάτων που δεν είναι στάσιμα είναι ο Μετασχηματισμός Κυματιδίων ή Wavelet Transform (WT), ο οποίος έχει αποδειχθεί ότι είναι πιο αποτελεσματικός στην χρήση επεξεργασίας σήματος ΗΕΓ όταν πρόκειται για διάγνωση ασθένειας [57].

Για την αφαίρεση του θορύβου, μία απλή και διαδεδομένη τεχνική είναι η εφαρμογή ζωνοπερατών φίλτρων, τα οποία επιτρέπουν την διέλευση μίας ζώνης συχνοτήτων και αποκόπτουν το υπόλοιπο σήμα. Ωστόσο, η τεχνική αυτή εμπεριέχει το μειονέκτημα ότι η αποκοπή ή όχι του σήματος γίνεται ανεξαρτήτως προέλευσης, δεν υπάρχει, δηλαδή, πρότερη γνώση για το αν είναι επιθυμητό σήμα ή αν είναι θόρυβος. Για τον λόγο αυτόν, η χρήση και άλλων τεχνικών είναι απαραίτητη με σκοπό να γίνει διαχωρισμός σήματος σε συνιστώσες [58], ώστε η μελέτη της προέλευσής τους να σταθεί δυνατή.

Η πλέον χαρακτηριστική τεχνική στον τομές αυτόν είναι η ανάλυση ανεξάρτητων συνιστωσών (Independent Component Analysis - ICA) [4],[58],[59],[60]. Η μέθοδος αυτή χρησιμοποιείται για τον διαχωρισμό ενός πολυμεταβλητού σήματος σε γραμμικό συνδυασμό συνιστωσών. Αυτό επιτυγχάνεται με την υπόθεση ότι οι συνιστώσες είναι, δυνητικά, μη γκαουσιανά σήματα και στατιστικά ανεξάρτητα

[61]. Χρησιμοποιείται τόσο για την εξάλειψη παραμορφώσεων, ανασυνθέτοντας το σήμα αφού αποκλειστούν οι πηγές που σχετίζονται με θόρυβο [58], καθώς και για τη ανάλυση των αλληλεπιδράσεων ανάμεσα σε διαφορετικές περιοχές του εγκεφάλου και μεταξύ σημάτων διαφορετικής φύσης όπως παραδείγματος χάριν προκλητού δυναμικού και δραστηριότητας υποβάθρου. Ωστόσο, η χρήση αυτής της τεχνικής συνοδεύεται από ορισμένους περιορισμούς[59]. Αρχικά, το μέγιστο πλήθος των συνιστωσών που μπορούν να βρεθούν ισούται με το πλήθος των ηλεκτροδίων που χρησιμοποιήθηκαν κατά την καταγραφή του σήματος, αλλά πέραν αυτού του περιορισμού, και η επιλογή του ακριβούς αριθμού των συνιστωσών και η ερμηνεία αυτών δεν είναι αντικειμενική. Επιπλέον, η εφαρμογή της μεθόδου προϋποθέτει μεγάλο όγκο δεδομένων και βασίζεται στην υπόθεση ότι οι πηγές των συνιστωσών παραμένουν σε σταθερές θέσεις μέσα στον εγκέφαλο, κάτι που δεν είναι πάντα ακριβές. Τέλος, όπως αναφέρθηκε προηγουμένως, η μέθοδος υποθέτει ότι οι συνιστώσες δεν ακολουθούν γκαουσιανή κατανομή.



Εικόνα 21 *Επεξήγηση της ανάλυσης ανεξάρτητων συνιστωσών (ICA)*

Όσον αφορά την ανάδειξη της σχετικής με το γεγονός δραστηριότητας, η μέθοδος που χρησιμοποιείται για την ανάλυση εξαρτάται με το είδος της [37]. Σημειώνεται ότι τα προκλητά δυναμικά αντιστοιχούν σε ερέθισμα από δραστηριότητα τόσο time-locked όσο και phase-locked, ενώ ο συγχρονισμός και ο αποσυγχρονισμός είναι μόνο time-locked ως προς το ερέθισμα. Η κοινή βάση που ενώνει τις δύο αυτές τεχνικές είναι ότι υπολογίζεται ο μέσος όρος [3],[33],[35],[37]. Η χρήση αυτού στηρίζεται στο ότι το συνολικό σήμα ηλεκτροεγκεφαλογραφήματος που καταγράφεται κατά τη διάρκεια της χορήγησης ενός ερεθίσματος εμπεριέχει τόσο την δραστηριότητα που σχετίζεται με το ερέθισμα όσο και αυτήν που είναι ασυσχέτιστη με αυτό και αποκαλείται δραστηριότητα υποβάθρου (background EEG), η οποία είναι αποτέλεσμα άλλων διεργασιών. Σημαντικό πρόβλημα, ωστόσο, αποτελεί το γεγονός ότι οι αλλαγές που είναι σχετικές με το ερέθισμα είναι της τάξης των μV, μικρότερες δηλαδή από το σύνηθες ηλεκτροεγκεφαλογραφικό σήμα, και για το λόγο αυτό προκαλούν πολύ μικρές μεταβολές στο σύνολο του σήματος. Επιπλέον, είναι πιθανό η δραστηριότητα που οφείλεται στο υπόβαθρο να κυμαίνεται στις ίδιες συχνότητες με αυτήν της απόκρισης, κρίνοντας το φιλτράρισμα ανώφελο. Ως εκ τούτου, υπάρχει η ανάγκη για μία τεχνική που να απορρίπτει την δραστηριότητα υποβάθρου και να αναδεικνύει την δραστηριότητα που σχετίζεται με την απόκριση στο ερέθισμα.

Η εφαρμογή του μέσου όρου, λοιπόν, βασίζεται στην υπόθεση ότι η δραστηριότητα του υποβάθρου δεν είναι time-locked με το γεγονός και θεωρεί ότι είναι ασυσχέτιστος με αυτό θόρυβος μηδενικής μέσης τιμής. Σε αντίθεση, το σήμα που αντιστοιχεί στην απόκριση στο ερέθισμα έχει συγκεκριμένη μορφή και είναι time-locked ως προς αυτό. Επομένως, βάσει αυτής της υπόθεσης, το εκάστοτε πείραμα επαναλαμβάνεται πολλαπλές φορές (δοκιμές – trials/epochs) ώστε ο υπολογισμός της μέσης

τιμής να απορρίψει εν τέλει την ασυσχέτιστη δραστηριότητα και να ενισχύσει τον σηματοθορυβικό λόγο. Ωστόσο, η τεχνική αυτή έρχεται με κάποιους περιορισμούς.

Αρχικά, δεν είναι πάντα εφικτή η εφαρμογή πολλαπλών επαναλήψεων της εργασίας ή η μεγάλη διάρκεια αυτών, επειδή η διαδικασία αυτή μπορεί να επιφέρει κόπωση στον εξεταζόμενο. Πιο συγκεκριμένα, πέραν της κούρασης εξαιτίας της εργασίας πρέπει να ληφθεί υπόψιν και η κούραση εξαιτίας της χρήσης της διάταξης καταγραφής [33]. Επιπλέον, ο εγκέφαλος δεν ανταποκρίνεται με ακριβώς τον ίδιο τρόπο στο ερέθισμα σε κάθε επανάληψη. Παραδείγματος χάριν, για ένα προκλητό δυναμικό προσδοκάται να υπάρχει μεταβολή τόσο στο πλάτος της απόκρισης όσο και στον λανθάνοντα χρόνο, εξαιτίας κούρασης, εξοικείωσης με το ερέθισμα κ.ά. Η αλλαγή κυρίως στον λανθάνοντα χρόνο μπορεί να αλλάξει σημαντικά το αποτέλεσμα του υπολογισμού του μέσου όρου [33],[59]. Ακόμα, η διαφοροποίηση της δραστηριότητας υποβάθρου όσον αφορά την χωρική και χρονική κατανομή μεταξύ των επαναλήψεων θέτει ανακριβή τη θεώρηση ότι πρόκειται για ασυσχέτιστο με το ερέθισμα θόρυβο. Οι μεταβολές είναι πιθανό να προκαλούνται από άλλες διεργασίες ή αλλαγές στην απόδοση και την γενικότερη κατάσταση του εξεταζόμενου, όπως για παράδειγμα συγκέντρωση, κόπωση, εξοικείωση κ.ά. Παραδείγματος χάριν, έχει δειχθεί ότι τα οπτικά ερεθίσματα μειώνουν το πλάτος του ηλεκτροεγκεφαλογραφήματος [59]. Τέλος προτείνεται η ίδια εργασίας να μην επαναλαμβάνεται περιοδικά, διότι σε περίπτωση που γίνει ο μέσος όρος μπορεί να περιλαμβάνει εκτός της προσδοκώμενης δραστηριότητας και άλλες διεργασίες του εγκεφάλου που μπορεί να είναι ασυσχέτιστες με το εκάστοτε ερέθισμα αλλά να είναι περιοδικές [33].

6.5.2. Ανάλυση Δεδομένων

Μετά την ανάλυση και προεπεξεργασία των δεδομένων που έχουν σαν σκοπό την εξαγωγή του καθαρού σήματος χωρίς παρεμβολές, εξίσου σημαντική αν όχι περισσότερο είναι η ερμηνεία του σήματος. Το ηλεκτροεγκεφαλογράφημα καταγράφει σήματα τα οποία μπορεί να προέρχονται από διάφορες εγκεφαλικές περιοχές και να ανήκουν σε διάφορες ζώνες συχνοτήτων, όπως αυτές περιγράφηκαν παραπάνω. Η γνώση των ιδιοτήτων του κάθε ρυθμού βοηθούν στην κατανόηση των διεργασιών που λαμβάνουν χώρα στον εγκέφαλο ανά πάσα στιγμή.

Πιο αναλυτικά, ο μειωμένος ρυθμός α στον προμετωπιαίο και ινιακό λοβό έχει συσχετιστεί με την επιτυχή κωδικοποίηση μνήμης [62], ενώ το ίδιο φαινόμενο στο οπίσθιο μέρος του κεφαλιού παρατηρείται στην επιτυχή συντήρηση της χωρικής εργαζόμενης μνήμης σε σχέση με την ανεπιτυχή, που οδηγεί σε λανθασμένες αποκρίσεις [63]. Πιο συγκεκριμένα, η επιτυχής κωδικοποίηση μνήμης έχει βρεθεί ότι συνοδεύεται από αυξημένο συγχρονισμό στις ζώνες θ στον δεξί μετωπιαίο φλοιό και γ στις βρεγματικές-ινιακές περιοχές, καθώς και από μειωμένο συγχρονισμό στις ζώνες β και α στον προμετωπιαίο και ινιακό φλοιό [62]. Επιστρέφοντας στη ζώνη α, έχει βρεθεί ότι οι αναπαραστάσεις που μοιράζονται μεταξύ εικόνων και αντίληψης προκύπτουν ειδικά σε αυτήν τη ζώνη. Αυτές οι αναπαραστάσεις είναι εμφανείς στα οπίσθια έναντι των εμπρόσθιων ηλεκτροδίων, υποδηλώνοντας μία προέλευση από τον βρεγματικό-ινιακό φλοιό [64]. Ωστόσο, ο κύριος ρόλος του ρυθμού α στην εργαζόμενη μνήμη φαίνεται να είναι η αναστολή των άσχετων ερεθισμάτων ή πιο γενικά, η αναστολή των εγκεφαλικών περιοχών που δεν είναι κρίσιμες με την σχετική τρέχουσα εργασία [63].

Όσον αφορά τον ρυθμό β, έχει συσχετιστεί με μετατοπίσεις της προσοχής, καθώς και με φιλτράρισμα άσχετων οπτικών αναπαραστάσεων [63]. Σχετικά με το οπτικό τμήμα του εγκεφάλου, οι πληροφορίες χωρίζονται σε οπίσθιας ανάδρασης (feedback) και πρόσθιας ανάδρασης (feedforward) και μεταφέρονται από διαφορετικά κανάλια νευρικών ταλαντώσεων. Έτσι οι ρυθμοί θ και γ μεταφέρουν πληροφορίες πρόσθιας ανάδρασης, ενώ οι ρυθμοί α και β οπίσθιας [64].

Οι ταλαντώσεις στο εύρος ζώνης γ πιστεύεται ότι αντανακλούν διαδικασίες που σχετίζονται με την ενεργοποίηση και τη διατήρηση νευρωνικών αναπαραστάσεων αντικειμένων [62]. Πειράματα έχουν

δείξει ότι ο ρυθμός γ αυξάνεται στα δεξιά πλάγια μετωπιαία σημεία κατά την συγκράτηση χωρικής θέσης και στον μετωπιαίο λοβό κατά τη διάρκεια συντήρησης της χωρικής μνήμης, σε αντίθεση με τη χρονική σειρά και χρονική εργαζόμενη μνήμη που ενισχύουν τις ζώνες θ, α και β [63]. Επιπλέον, κατά την κωδικοποίηση οπτικών ερεθισμάτων τα οποία μετά οι συμμετέχοντες θυμούνται, η μετωπιαία φάση θ και η οπίσθια ισχύς γ ενισχύονται, σε σχέση με εκείνα τα οποία στη συνέχεια οι συμμετέχοντες ξεχνούν [62]. Όσον αφορά την εργαζόμενη μνήμη, το 1995 οι Lisman και Idiart πρότειναν ένα μοντέλου του ρόλου των αλληλεπιδράσεων μεταξύ των δύο αυτών ζωνών συχνοτήτων - γ και θ-. Είπαν ότι τα αντικείμενα αναπαριστώνται στον φλοιό από διαδοχικούς κύκλους γ, οι οποίοι είναι εμφωλευμένοι μέσα σε δεδομένο κύκλο θ και ότι οι περιορισμοί στην χωρητικότητα της βραχυπρόθεσμης μνήμης προκύπτουν ως συνέπεια του αριθμού των κύκλων γ που χωρούν σε έναν κύκλο θ [65].

Αναφορικά με τον ρυθμό δ, στον οποίο ανήκουν τα πιο μικρά εγκεφαλικά κύματα που καταγράφονται στον άνθρωπο, με βάση θεωρητικές εκτιμήσεις και εμπειρικά ευρήματα, είναι πιθανό να διαδραματίζει καθοριστικό ρόλο στην επικοινωνία μεγάλης κλίμακας μεταξύ των περιοχών του εγκεφάλου [62].

Τέλος, η ζώνη θ κυρίως παρατηρείται κατά τον χωρικό προσανατολισμό, την πλοήγηση και την ανάκτηση χωρικών πληροφοριών. Πιο συγκεκριμένα, δύο όχι αμοιβαία αποκλειόμενες υποθέσεις έχουν προταθεί: οι ταλαντώσεις χαμηλής συχνότητας θ (~4Hz) συντονίζουν αισθητηριακές και κινητικές περιοχές κατά τη διάρκεια της πλοήγησης ή παίζουν ρόλο στη χωρική κωδικοποίηση και τη μνήμη [18]. Ταλαντώσεις θ έχουν παρατηρηθεί στον έσω κροταφικό λοβό (MTL) κατά τη διάρκεια της πλοήγησης σε σχέση με την ακινησία και συγκεκριμένα ενισχυμένες κατά τη διάρκεια πλοήγησης με στόχο, σε σύγκριση με την άσκοπη περιπλάνηση, γεγονός που υποδηλώνει εμπλοκή της ζώνης θ στην ανάκτηση της χωρικής μνήμης και στο σχεδιασμό διαδρομής [18]. Το ίδιο φαινόμενο παρατηρείται και κατά την έναρξη της κίνησης για μεγάλο χρονικό διάστημα σε σύγκριση με το σύντομο, διαδρομές, δηλαδή, που απαιτούν ανάκτηση περισσότερων χωρικών πληροφοριών και σχεδιασμό μεγαλύτερης διαδρομής προκαλούν αυξημένη ισχύ στην ζώνη θ [18]. Τέλος, στα οικεία περιβάλλοντα δημιουργείται υψηλότερη ισχύς θ, φαινόμενο που μπορεί να σχετίζεται με το γεγονός ότι υπάρχουν περισσότερες αποθηκευμένες χωρικές πληροφορίες που πρέπει να ανακτηθούν για να καθοδηγήσουν την κίνηση [18]. Επιπλέον, εκτός από το ρυθμό θ, αυξημένη παρουσία που σχετίζεται με την πλοήγηση εμφανίζεται στις ιπποκαμπικές και παρα-ιποκαμπικές περιοχές στους ρυθμούς α, β και γ [18]. Συνολικά, τα παραπάνω αποτελέσματα υποδηλώνουν ότι οι ταλαντώσεις θ εμπλέκονται εκτός από τις χαμηλού επιπέδου αισθητηριακές και κινητικές διεργασίες και στην ανάκτηση χωρικών πληροφοριών.

Η αντιστοίχιση των διαφόρων εγκεφαλικών συχνοτήτων με συγκεκριμένες εγκεφαλικές διεργασίες και λειτουργίες είναι δύσκολη. Παρόλο που το μεγαλύτερο μέρος της πειραματικής βιβλιογραφίας συμφωνεί σε έναν περιορισμένο αριθμό αντιστοιχιών, η ολοκληρωτική χαρτογράφηση και κατανόηση των περίπλοκων λειτουργιών του εγκεφάλου και του τρόπου με τον οποίο αυτές διεκπεραιώνονται παραμένει μυστήριο.

7. Πειραματικό Πρωτόκολλο

7.1. Σκοπός Πειράματος

Σκοπός του παρόντος πειράματος είναι η μελέτη της εγκεφαλικής δραστηριότητας που σχετίζεται με τη χωρική μνήμη και τον χωρικό προσανατολισμό υπό τη συνθήκη παρεμπόδισης όρασης. Πιο αναλυτικά, στόχος είναι η μελέτη της διαφοροποίησης στην εγκεφαλική λειτουργία και δραστηριοποίηση κατά την κωδικοποίηση ενός χώρου στον εγκέφαλο όταν αυτή αξιοποιεί την όραση και τα οπτικά ερεθίσματα σε σχέση με την απουσία αυτών. Επιπλέον, ζητούμενο είναι η μελέτη της αλλαγής στην εγκεφαλική λειτουργία ενός ατόμου κατά τον προσανατολισμό και την εκτέλεση εργασιών σε ένα ήδη γνωστό χώρο όταν αυτός έχει κωδικοποιηθεί με και χωρίς την βοήθεια της όρασης.

7.2. Προηγούμενα Ευρήματα

Ο τομέας της μνήμης, της χωρικής αναπαράστασης και του χωρικού προσανατολισμού άρχισαν να μελετώνται εκτενώς κάποια χρόνια πριν και με την συνεχή βελτίωση της τεχνολογίας που έχει ως αποτέλεσμα τη συνεχή βελτίωση του εξοπλισμού πολλές διατυπώσεις και επαληθεύσεις έχουν γίνει σε αυτούς του τομείς. Οι συχνότητες άλφα (8 – 13 Hz) και θήτα (4 – 8 Hz) αποτελούν τις πιο εκτενώς μελετημένες ζώνες συχνοτήτων σε αυτούς τους τομείς καθώς έχει επανειλημμένα αποδειχθεί ότι σχετίζονται με τις νοητικές καταστάσεις, στρατηγικές και ερεθίσματα. Σημειώνεται ότι στη βιβλιογραφία που μελετήθηκε, δεν βρέθηκαν ταυτόσημα πειράματα με αυτό που αναπτύχθηκε στο πλαίσιο της παρούσας εργασίας, οπότε παρατίθενται ευρήματα από μελέτες που σχετίζονται γενικότερα με τα παραπάνω πεδία.

Η δραστηριότητα στη ζώνη θήτα συσχετίζεται άμεσα με την προσπάθεια διατήρησης της μνήμης και αυξάνεται με την δυσκολία του έργου που ο συμμετέχοντας στο εκάστοτε πείραμα καλείται να φέρει εις πέρας. Στο πείραμα των Jensen και Tensen [66] από τους συμμετέχοντες ζητήθηκε να απομνημονεύσουν λίστες από 1, 3, 5 ή 7 οπτικά παρουσιαζόμενα ψηφία μέσα σε 3 δευτερόλεπτα. Κατά τη διάρκεια της απομνημόνευσης παρατηρήθηκε δραστηριότητα θήτα ζώνης, κυρίως των συχνοτήτων 7 - 8.5 Hz, που καταγράφηκε στις μετωπιαίες περιοχές του εγκεφάλου. Η δραστηριότητα αυτή αυξήθηκε παραμετρικά με τον αριθμό των στοιχείων που οι συμμετέχοντες έπρεπε να διατηρήσουν στην εργαζόμενη μνήμη τους (working memory) και διατηρήθηκε και στην περίοδο διατήρηση της μνήμης αλλά και στην ανάκτηση αυτής. Παρόμοια αποτελέσματα βρέθηκαν και από μελέτη με ηλεκτροεγκεφαλογράφο από το Gevins [67].

Από την άλλη μεριά, η δραστηριότητα στη ζώνη συχνοτήτων άλφα έχει αρκετούς λειτουργικούς συσχετισμούς με τις λειτουργίες των αισθήσεων, της κίνησης και της μνήμης. Η συχνότητα άλφα παρατηρείται κυρίως στο πίσω μέρος του εγκεφάλου, συμπεριλαμβανομένων των ινιακών, βρεγματικών και οπίσθιων κροταφικών περιοχών. Η μείωση της ζώνης άλφα μεταφράζεται σε ενεργοποίηση του ατόμου ή γνωστική ετοιμότητα του φλοιού για την επεξεργασία νέων πληροφοριών [68]. Στην έρευνα των Sauseng και συνεργατών όπου οι συμμετέχοντες καλούνταν να στρέψουν την προσοχή τους στην δεξιά ή αριστερή πλευρά του χώρου για να παρατηρήσουν κάτι, βρέθηκε ότι η ζώνη άλφα παρουσίασε διακυμάνσεις ανάλογες με την προσοχή. Τόσο οι χαμηλές όσο και οι υψηλές ζώνες άλφα βρέθηκε να καταστέλλονται περισσότερο σε θέσεις αντίθετες με τον ημιπέδιο (διαχωρισμός του πεδίου που έχει μπροστά του ο χρήστης σε δύο μισά -δεξιά και αριστερά-)το οποίο το άτομο παρακολουθούσε [69].

Σχετικά με τον χωρικό προσανατολισμό, στο πείραμα του Plank και των συνεργατών [68] αποδείχθηκε ότι αναλόγως με το πλαίσιο αναφοράς που επιλέγει ο κάθε συμμετέχοντας, δηλαδή αν θα

χρησιμοποιήσει εγωκεντρικό ή αλλοκεντρικό πλαίσιο, εντοπίζονται διαφορές στην φασματική ενεργοποίηση της ζώνης άλφα εντός ή κοντά σε ινιακές, ινιακοβρεγματικές καθώς και οπίσθιους βρεγματικές περιοχές. Αυτές οι διαφορές μπορεί να συνδέονται με τις διαφορετικές ανάγκες ενημέρωσης των εγωκεντρικών και αλλοκεντρικών πλαισίων κάθε φορά σε σχέση με τη νέα θέση του σώματος ως προς το περιβάλλον του. Αντίθετα και οι δύο ομάδες συμμετεχόντων εμφάνισαν συγκρίσιμα και διαμορφωμένα με βάση την πολυπλοκότητα μοτίβα ενεργοποίησης της ζώνης συχνοτήτων θήτα. Αυτά ήταν μέσα ή κοντά στον μέσο μετωπιαίο φλοιό, πιθανότητα αντικατοπτρίζοντας την αυξημένη γνωστική προσπάθεια.

Επιπροσθέτως, ο τομέας της απουσίας όρασης και οι αλλαγές που αυτή επιφέρει στον εγκέφαλο και τις διεργασίες αυτού έχει μελετηθεί εκτεταμένα. Η όραση είναι ίσως η σημαντικότερη από τις βασικές αισθήσεις του ανθρώπου και το μέσο με το οποίο αντιλαμβάνεται τον κόσμο στο μεγαλύτερο μέρος του.

Η μελέτη του Santaniello και των συνεργατών [70] μελέτησε τις επιπτώσεις της παρεμποδισμένης όρασης σε άτομα με κατά τα άλλα κανονική όραση, εμποδίζοντας την όραση στους συμμετέχοντες για δύο ώρες. Στη συνέχεια, μέσω ηλεκτροεγκεφαλογραφήματος κατέγραψαν μειωμένη ένταση της ζώνης συχνοτήτων άλφα στα άτομα με παρεμποδισμένη όραση σε σύγκριση με αυτά χωρίς. Αυτό μπορεί να αντανακλά στην στρατολόγηση περισσότερων πόρων με σκοπό το άτομο να μπορεί να ανταπεξέλθει στις απαιτήσεις του πειράματος. Αυτή η μείωση είναι συνεπής με τόσο τη μειωμένη ανασταλτική δραστηριότητα όσο και με την αυξημένη γνωστική επεξεργασία που παρατηρήθηκε από τα πειράματα των Klimesch, Sauseng και Hanslmayr [71]. Μειωμένη δραστηριότητα στην άλφα ζώνη έχει επίσης παρατηρηθεί σε τυφλά άτομα στις βρεγματικό-ινιακό περιοχές του εγκεφάλου και έχει συσχετιστεί με αυξημένη διεγερσιμότητα του οπτικού φλοιού [72].

Μία ακόμα μελέτη που έλαβε χώρα για να μελετηθούν οι διαφορές στην τοπολογία του ηλεκτροεγκεφαλογραφήματος μεταξύ των ατόμων με κανονική όραση και των ατόμων με τύφλωση εκ γενετής είναι αυτή της Annes και των συνεργατών [73]. Το πείραμα αυτό περιείχε εργασίες σχετικές με την απτική αντίληψη με την χωρική απεικόνιση. Και στις δύο αυτές εργασίες η δραστηριότητα στο πάνω άκρο της ζώνης συχνοτήτων άλφα ήταν σημαντικά χαμηλότερη στα άτομα με τύφλωση σε σύγκριση με τα άτομα κανονικής όρασης στις βρεγματικό-ινιακές περιοχές του εγκεφάλου. Ίδια τάση εμφάνισαν και οι χαμηλότερες συχνότητες της άλφα. Συγκεντρωτικά οι διαφοροποιήσεις στην άλφα ζώνη ήταν πιο έντονες στις δύο οπίσθιες συστάδες ηλεκτροδίων που καλύπτουν περιοχές του εγκεφάλου στις οποίες κυριαρχεί το οπτικό σύστημα στα υγιή άτομα (O1-O2). Το γεγονός ότι και στα δύο πειράματα υπήρχε μείωση της κυρίως άνω ζώνης άλφα στο οπίσθιο μέρος του κρανίου, σημαίνει ότι οι αλλαγές αυτές στους εκ γενετής τυφλούς δεν εξαρτώνται άμεσα με τις τρέχουσες αισθητηριακές ή γνωστικές διεργασίες, αλλά είναι μάλλον ανεξάρτητες. Παρατηρήθηκε επίσης ότι στα υγιή άτομα οι συχνότητες άλφα μειώνονται όταν ένα αισθητηριακό ερέθισμα συμβαίνει ή το γνωστικό φορτίο αυξάνεται. Το φαινόμενο αυτό του αποσυγχρονισμού είναι γνωστό ως ‘παρεμπόδιση της άλφα’ (alpha blocking) [74]. Υπήρχαν ακόμα ενδείξεις ότι η άνω και κάτω ζώνη άλφα συνδέονται με διάφορες αντιληπτικές και γνωστικές λειτουργίες. Μπορεί να υποτεθεί ότι η μείωση της δραστηριότητας της άνω ζώνης άλφα στο οπίσθιο κρανίο στους τυφλούς αντανακλά μία ισχυρότερη ενεργοποίηση πολυαισθητηριακών και οπτικών περιοχών του εγκεφάλου κατά τη διάρκεια μη οπτικών αντιληπτικών και γνωστικών λειτουργιών στους τυφλούς παρά στους βλέποντες. Αυτή η υπόθεση συνάδει με την ιδέα της λειτουργικής αναδιοργάνωσης των υποβαθμισμένων «οπτικών» περιοχών.

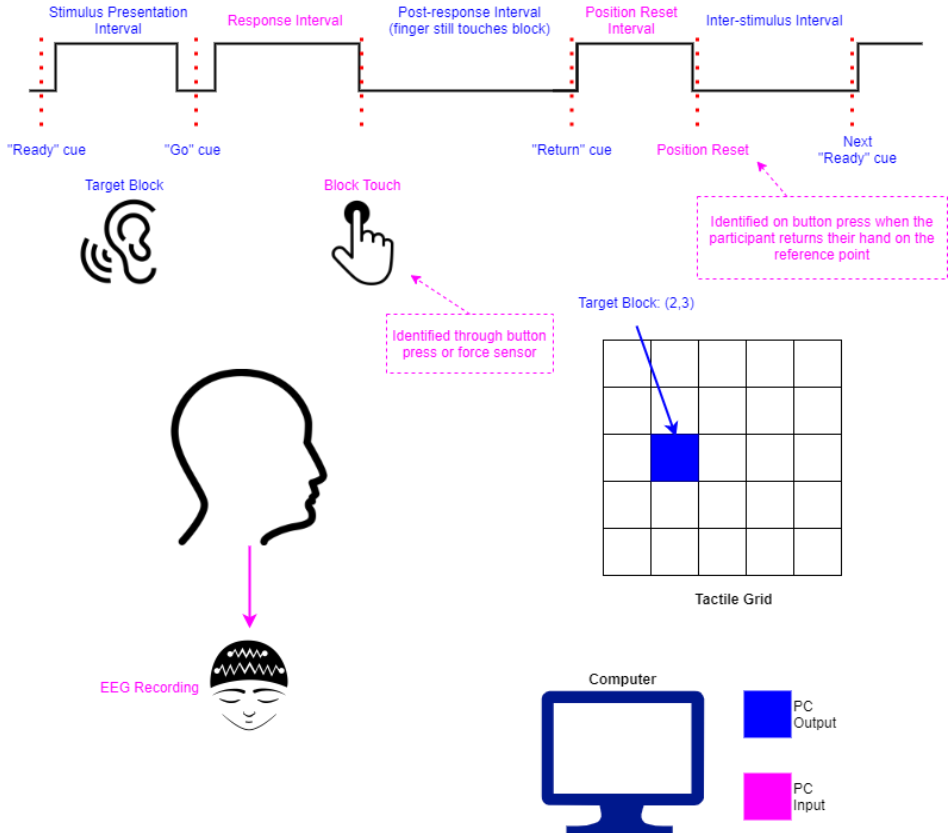
Επίσης, έχουν γίνει πειράματα όπου η όραση σε υγιή άτομα παρεμποδίζεται ασυμμετρικά. Για την ακρίβεια στο πείραμα του Levick και των συνεργατών [75] όπου μέσω ειδικών φακών εμπόδισαν την όραση μόνο στο ένα μάτι. Το πείραμα αυτό επιβεβαίωσε την αρχική τους υπόθεση ότι η ασυμμετρία στην οπτική παρεμπόδιση θα έχει ως αποτέλεσμα και ασυμμετρία στο ηλεκτροεγκεφαλογράφημα. Η

υπόθεση αυτή βασίστηκε στην ασυμμετρική μείωση της ετοιμότητας και της ημισφαιριακής επαγρύπνησης που θα έχει ως αποτέλεσμα αυξημένη θήτα, τον ρυθμό του ηλεκτροεγκεφαλογραφήματος που συνδέεται άμεσα με υπνηλία και υπναγωγικές καταστάσεις. Πράγματι, το ημισφαίριο στο οποίο η όραση είχε παρεμποδιστεί παρήγαγε περισσότερες συχνότητες στη ζώνη θήτα. Παρόμοιες αλλαγές έχουν παρατηρηθεί κατά τη διάρκεια αισθητηριακής απομόνωσης όπου ο ρυθμός θήτα αυξάνεται ενώ αντίθετα ο ρυθμός άλφα μειώνεται μέσα σε διάστημα ημερών [76].

Όπως γίνεται ξεκάθαρο από όλα τα παραπάνω παραδείγματα τόσο η χωρική μνήμη και προσανατολισμός όσο και η παρεμπόδιση της όρασης και οι επιπτώσεις αυτής στο ηλεκτροεγκεφαλογράφημα έχουν μελετηθεί εκτενώς. Ωστόσο, με βάση την γνώση του συγγραφέως μέχρι αυτήν τη στιγμή, καμία έρευνα δεν έχει γίνει όπου συγκρίνονται τα αποτελέσματα του ηλεκτροεγκεφαλογράφου κατά τη διάρκεια προσανατολισμού σε έναν χώρο χωρίς την βοήθεια της όρασης όταν αυτός ο χώρος έχει κωδικοποιηθεί για πρώτη φορά με την βοήθεια μόνο της όρασης σε αντίθεση με την κωδικοποίηση για πρώτη φορά μόνο με την αφή. Επομένως άμεση σύγκριση με ήδη υπάρχοντα αποτελέσματα για αυτό το πείραμα δεν μπορούν να συμβούν.

7.3. Περιγραφή Πειράματος

Η παρακάτω εικόνα παρουσιάζει περιληπτικά τη βασική δομή του πειράματος που υλοποιήθηκε και πραγματοποιήθηκε στα πλαίσια αυτής της εργασίας. Αρχικά, η γενική ιδέα του πειράματος είναι η μελέτη του προσανατολισμού σε έναν χώρο χωρίς την βοήθεια της όρασης όταν αυτός ο χώρος κωδικοποιείται για πρώτη φορά με την βοήθεια μόνο της όρασης σε αντίθεση με την κωδικοποίηση για πρώτη φορά μόνο με την αφή. Για τον λόγο αυτόν, οι συμμετέχοντες χωρίστηκαν σε δύο ομάδες, μία με την όραση της ανεμπόδιστη κατά τη διάρκεια της κωδικοποίησης του χώρου αλλά παρεμποδισμένη στη συνέχεια κατά τη διάρκεια της πλοήγησης, και μία με καλυμμένα τα μάτια καθ' όλη τη διάρκεια του πειράματος.



Εικόνα 22 *Σχηματική αναπαράσταση του πειραματικού πρωτοκόλλου*

Στο πάνω μέρος της εικόνας, διακρίνεται με γραμμικό τρόπο η χρονική εξέλιξη του κυρίως μέρους του πειράματος χωρισμένη σε όλα τα διαφορετικά στάδια από τα οποία πέρασε ο κάθε εξεταζόμενος, ανεξαρτήτως σε ποια ομάδα ανήκε. Αυτό το κύριο μέρος ήταν ένας επαναληπτικός βρόγχος. Πριν το κυρίως μέρος οι συμμετέχοντας περνούσαν από τρία επιπλέον στάδια με σκοπό της εξαγωγή δεδομένων σύγκρισης ή αλλιώς το σημείο αναφοράς, τα οποία ωστόσο θα αναλυθούν παρακάτω.

Η αρχή του βασικού μέρους γίνεται από το 'Ready cue' ή αλλιώς έναν χαρακτηριστικό ήχο 'μπιπ' που σηματοδοτεί την έναρξη. Στην συνέχεια, ακολουθεί το Stimulus Presentation Interval, δηλαδή η ηχητική περιγραφή του σημείου του χώρου που ο συμμετέχοντας καλείται να εντοπίσει, πράγμα το οποίο και πρέπει να κάνει μετά το ηχητικό σήμα 'Go cue' το οποίο είναι επίσης ένας ήχος 'μπιπ'. Η κίνηση του χεριού του συμμετέχοντα και εν τέλει η κατάληξη στο σημείο του χώρου που του παρουσιάστηκε πριν, λαμβάνει χώρα εντός του χρονικού πλαισίου 'Response Interval'. Το χρονικό διάστημα 'Response Interval' τερματίζεται με την επαφή του σημείου και ο συμμετέχοντας συνεχίζει να αγγίζει το ζητούμενο σημείο στο διάστημα 'Post-response Interval' μέχρι την ήχηση του σήματος 'Return cue' -πάλι ηχητικό 'μπιπ'- όπου πρέπει το χέρι να επιστραφεί στην θέση εκκίνησης, μία ουδέτερη θέση εκτός του χώρου άλλα προκαθορισμένη και πάντα η ίδια (Position Reset Interval). Όταν το χέρι αγγίζει το σημείο αναφοράς κατά την χρονική στιγμή 'Position Reset'υπάρχει ένα διάστημα κενό, το 'Inter-stimulus Interval', στο οποίο ο συμμετέχοντας τερματίζει αυτήν την επανάληψη του βασικού πειράματος και προετοιμάζεται για την επόμενη επανάληψη, με τον ήχο 'Next "Ready" cue' να σηματοδοτεί και επίσημα την έναρξη του επόμενου κύκλου.

Το σύνολο του πειράματος σχεδιάστηκε και υλοποιήθηκε με την βοήθεια του PsychoPy και συγκεκριμένα την έκδοση 2022.2.5. Το PsychoPy είναι ένα λογισμικό ανοιχτού κώδικα γραμμένο στην γλώσσα προγραμματισμού Python και χρησιμοποιείται ευρέως κυρίως στη νευροεπιστήμη και την πειραματική ψυχολογία. Το συγκεκριμένο πρόγραμμα υποστηρίζει την συγγραφή κώδικα για την δημιουργία πειραμάτων, ωστόσο σε αντίθεση με τα περισσότερα πακέτα, δίνει και την δυνατότητα χρήσης μίας ειδικά σχεδιασμένης γραφικής διεπαφής (UI), χαρακτηριστικό που αξιοποιήθηκε για την κατασκευή του παρόντος πειράματος.

Για την καταγραφή της εγκεφαλικής δραστηριότητας, από τους συμμετέχοντες του πειράματος ζητήθηκε κατά τη διάρκεια του συνόλου του πειράματος να φορούν ένα κράνος ηλεκτροεγκεφαλογράφου, το οποίο περιγράφεται στην επόμενη υποενότητα.

Μπαίνοντας σε περισσότερη ανάλυση, το πείραμα αποτελείται εκτός από το βασικό μέρος, το οποίο είναι σε μορφή βρόχου όπως αναφέρθηκε παραπάνω, και από άλλα τρία μέρη που λαμβάνουν χώρα στην αρχή και αφορούν στη λήψη απαραίτητων σημάτων που θα επιτρέψουν τη σύγκριση των δεδομένων και τη σωστή αξιολόγηση του κυρίως μέρους της πειραματικής διαδικασίας που λαμβάνει χώρα στο τέταρτο μέρος. Οι συμμετέχοντες κάθονται μπροστά από την οθόνη του υπολογιστή που είναι υπεύθυνος για την ροή του πειράματος και έχουν μπροστά τους σε μικρή απόσταση από τα χέρια τους τον χώρο που καλούνται να κωδικοποιήσουν.

Εικόνα 23 *Σχηματική αναπαράσταση των φάσεων του πειράματος. Με κίτρινο οι οδηγίες της κάθε φάσης και με κόκκινο η πειραματική διαδικασία*

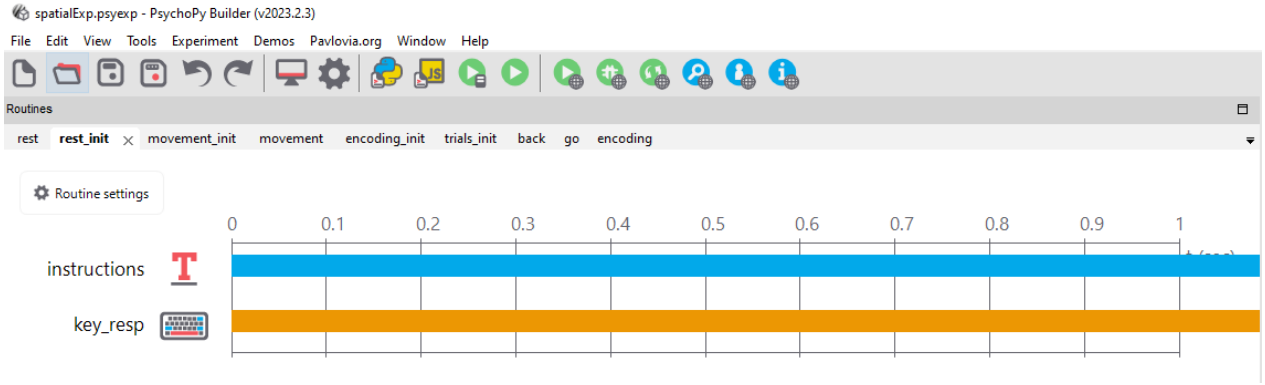
Η διαφορά ανάμεσα στις δύο ομάδες στις οποίες οι συμμετέχοντες χωρίστηκαν είναι ότι στην Ομάδα 1 τέθηκε να κωδικοποιήσει τον χώρο και να ολοκληρώσει τα πρώτα τρία μέρη του πειράματος με τη βοήθεια της όρασης, με σκοπό να δράσει ως μέτρο σύγκρισης για την Ομάδα 2. Από την άλλη, η

Ομάδα 2 ολοκλήρωσε το σύνολο του πειράματος με απουσία όρασης, με τους συμμετέχοντες της ομάδας να έχουν καλυμμένα τα μάτια τους με μία μάσκα συνεχώς.

Ο χώρος ο οποίος κλήθηκαν να κωδικοποιήσουν οι συμμετέχοντες ήταν ένα πλέγμα τετραγώνων 4x6 η κατασκευή του οποίου θα αναλυθεί αργότερα.

Κατά την έναρξη του πειράματος στην οθόνη του υπολογιστή που έβλεπαν οι συμμετέχοντες εμφανίστηκε ένα μήνυμα με τις οδηγίες για την πρώτη φάση της διαδικασίας. Όταν ήταν έτοιμοι και είχαν κατανοήσει πλήρως τις οδηγίες πατούσαν το πλήκτρο 'space' που τους μετέφερε αυτομάτως στο πρώτο μέρος του πειράματος. Αντίστοιχη οθόνη με οδηγίες εμφανίζεται στους συμμετέχοντες πριν την έναρξη κάθε μίας από τις υπόλοιπες 3 φάσεις του πειράματος. Η Ομάδα 1, επειδή είχε ανοιχτά τα μάτια διάβαζε κανονικά τις οδηγίες ενώ στην Ομάδα 2 οι οδηγίες διαβάζονταν.

Στην κάτω οθόνη διακρίνεται η υλοποίηση της οθόνης οδηγιών του πειράματος στο λογισμικό PsychoPy. Επιλέχθηκε ένα στοιχείο κειμένου (instructions) στο οποίο εμπεριέχεται το κείμενο με τις οδηγίες προς τους συμμετέχοντες για να εμφανίζεται στην οθόνη με την επιλογή να τερματίζεται μόνο όταν το στοιχείο key_resp αντιληφθεί το πάτημα του πλήκτρου space. Το κείμενο των οδηγιών περιέχει δύο τμήματα οδηγιών, ένα για την κάθε Ομάδα. Και τα δύο αυτά στοιχεία είναι ενεργά επ' αόριστον αν δεν πατηθεί το πλήκτρο space με σκοπό να δώσουν στους συμμετέχοντες όλον τον χρόνο που χρειάζονται. Μία τέτοια ίδια υλοποίηση υπήρχε στην αρχή πριν την έναρξη κάθε φάσης του πειράματος έτσι ώστε οι συμμετέχοντες να είναι πάντα σίγουροι και πλήρως ενημερωμένοι για το τί πρόκειται να συμβεί στη συνέχεια και τί ζητείται από τους ίδιους. Με την επιλογή αυτής της τμηματοποίησης αποφεύχθηκε τυχόν σύγχυση που μπορεί να προκαλούσαν οι ολικές οδηγίες στην αρχή του πειράματος για όλες τις φάσεις αυτού.

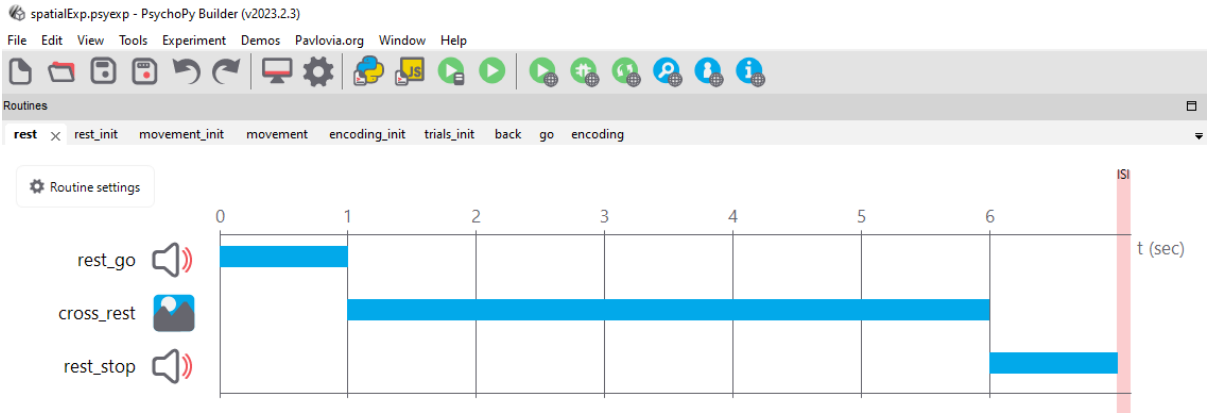


Εικόνα 24 *Δείγμα του τρόπου προβολής των οδηγιών. Ίδια διαδικασία πριν από κάθε φάση απλώς με διαφορετικό κείμενο*

Στο πρώτο μέρος του πειράματος, βασικός σκοπός ήταν η καταγραφή της δραστηριότητας του εγκεφάλου όταν το άτομο βρίσκεται σε κατάσταση ηρεμίας. Οι συμμετέχοντες της Ομάδας 1 εστίαζαν την προσοχή τους σε έναν μαύρο σταυρό που εμφανιζόταν στο κέντρο της οθόνης, ενώ οι συμμετέχοντες της Ομάδας 2 παρέμεναν με κλειστά τα μάτια για 5 δευτερόλεπτα. Και από τις δύο ομάδες είχε ζητηθεί να προσπαθήσουν να παραμείνουν σε όσο το δυνατόν μεγαλύτερη κατάσταση ηρεμίας κατά τη διάρκεια αυτής της φάσης. Τα σήματα της έναρξης και λήξης της περιόδου ηρεμίας (rest) δίνονταν με έναν χαρακτηριστικό ήχο 'μπιπ' και η διάρκεια της ηρεμίας διαρκούσε 5 δευτερόλεπτα πριν ακουστεί πάλι ο χαρακτηριστικός ήχος της λήξης.

Στην εικόνα που ακολουθεί απεικονίζεται η υλοποίηση αυτήν την φάσης στο λογισμικό ανοιχτού κώδικα. Ο ήχος 'μπιπ' είναι ένα αρχείο τύπου wav που έχει προστεθεί εντός ενός στοιχείου ήχου (rest_go) το οποίο διαρκεί ακριβώς ένα δευτερόλεπτο. Κατευθείαν μετά την λήξη του στοιχείου ήχου προβάλλεται η εικόνα του κεντραρισμένου μαύρου σταυρού που εμπεριέχεται στο στοιχείο της

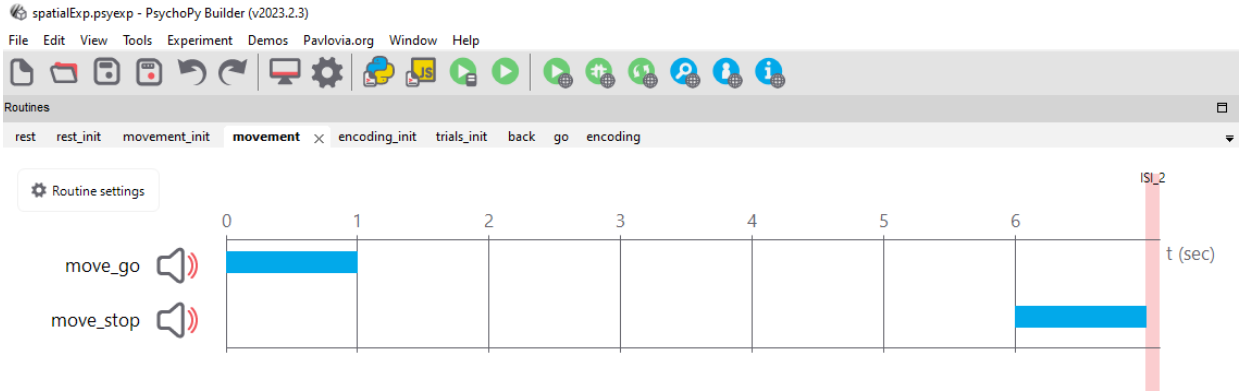
εικόνας cross_rest και διαρκεί για 5 δευτερόλεπτα πριν ξεκινήσει ένας πανομοιότυπος ήχος με την αρχή αυτής της φάσης με το όνομα rest_stop με τον οποίο και τελειώνει η πρώτη φάση. Ο διαχωρισμός ανάμεσα στις δύο ομάδες δεν ήταν απαραίτητος καθώς η Ομάδα 2 με παρεμποδισμένη όραση δεν μπορούσε να δει την εικόνα του σταυρού.



Εικόνα 25 Φάση 1: *Ο ήχος της έναρξης , η προβολή της εικόνας του κεντραρισμένου μαύρου σταυρού και ο ήχος της λήξης*

Στο δεύτερο μέρος, σκοπός ήταν να καταγραφεί η τυχαία κίνηση του χεριού του κάθε συμμετέχοντα με σκοπό στην συνέχεια να συγκριθεί με την κίνησή του κατά τη διάρκεια του κυρίως πειράματος. Για τον λόγο αυτό, και οι δύο Ομάδες κλήθηκαν να κλείσουν τα μάτια τους για 5 δευτερόλεπτα και να κινήσουν το δεξί τους χέρι τυχαία πάνω στον χώρο. Πάλι οι ήχοι έναρξης και λήξης πλαισίωναν αυτήν τη χρονική περίοδο.

Στην υλοποίηση του PsychoPy που υπάρχει στην συνέχεια υπάρχουν οι ίδιοι ήχοι έναρξης (move_go) και λήξης (move_stop) της φάσης, διάρκειας ενός δευτερολέπτου οι οποίοι πλαισιώνουν 5 δευτερόλεπτα όπου δεν υπάρχει κανένα στοιχείο στην οθόνη του υπολογιστή. Στο τέλος η κόκκινη ζώνη σηματοδοτεί τη λήξη αυτού του τμήματος.

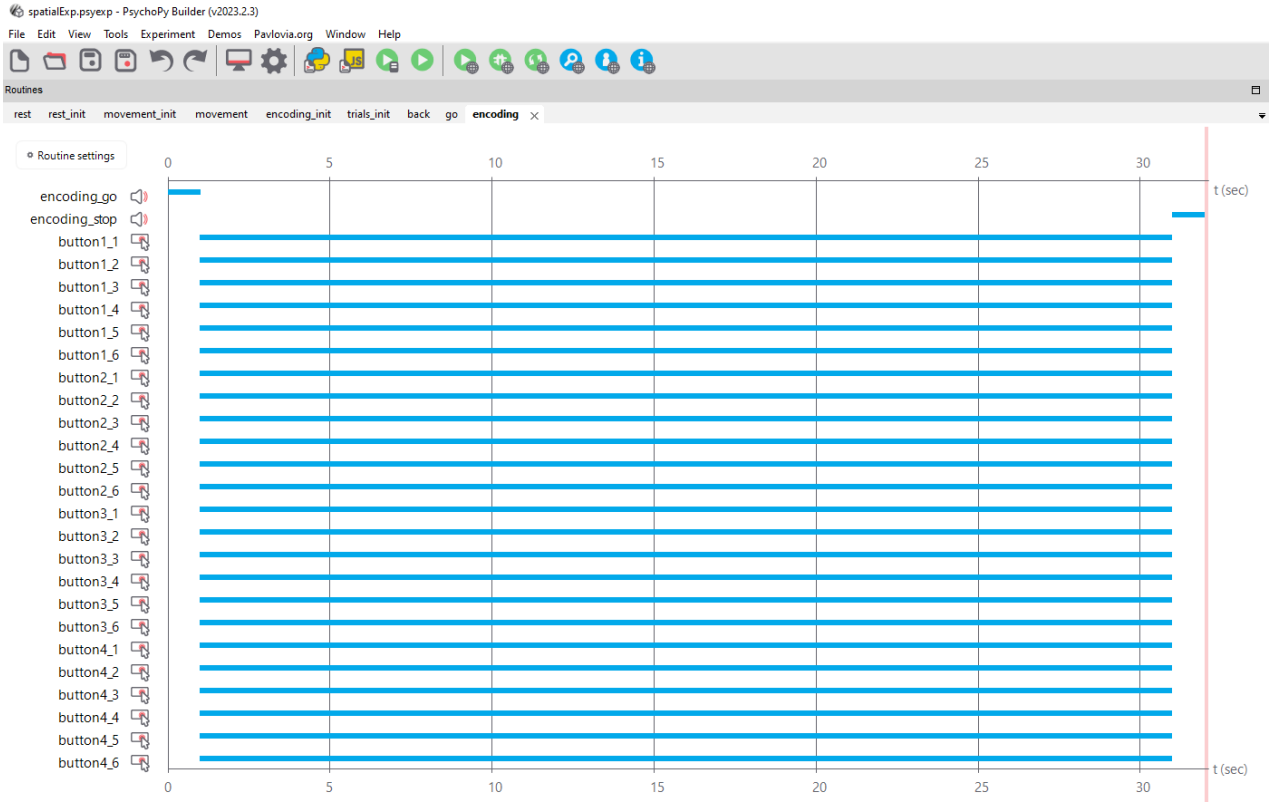


Εικόνα 26 Φάση 2: Ο *ήχος της έναρξης, παύση 5 δευτερολέπτων και ο ήχος της λήξης*

Το τρίτο μέρος αφορά στη διαδικασία της κωδικοποίησης του χώρου και στην οθόνη που βλέπουν οι συμμετέχοντες εμφανίζεται το πλέγμα 4x6. Κατά τη διάρκεια αυτής της φάσης, που διαρκούσε 30 δευτερόλεπτα, οι συμμετέχοντες της Ομάδας 1 έχοντας ανοιχτά τα μάτια και ακίνητο το κεφάλι κλήθηκαν να προσπαθήσουν να προσανατολιστούν στις διάφορες θέσεις της οθόνης χωρίς να κινήσουν τα χέρια τους. Από την άλλη πλευρά, η Ομάδα 2 με κλειστά τα μάτια και επίσης ακίνητο κεφάλι έπρεπε να κινήσει το κυρίαρχο χέρι πάνω στο πλέγμα του χώρου από αριστερά προς τα δεξιά και από πάνω προς τα κάτω ανά γραμμή με σκοπό την κωδικοποίησή του. Για να σταθεί δυνατό αυτό, πριν την έναρξη αυτού του μέρους για τους συμμετέχοντες της Ομάδας 2, πάνω από στον χώρο τοποθετούταν ανάγλυφο πλέγμα που χώριζε με ακρίβεια το πλέγμα του χώρου το οποίο στη συνέχεια

αφαιρούνταν. Τα ηχητικά σήματα που υποδήλωναν ότι οι συμμετέχοντες έπρεπε να ξεκινήσουν και να σταματήσουν την διαδικασία της κωδικοποίησης πλαισίωναν αυτά τα 30 δευτερόλεπτα.

Στην παρακάτω εικόνα διακρίνονται τα δύο πρώτα στοιχεία ήχου που είναι πάλι τα ηχητικά σήματα που πλαισιώνουν αυτήν την τρίτη φάση του πειράματος (encoding_go και encoding_stop). Στην συνέχεια υπάρχουν 24 στοιχεία πλήκτρων (button1_1 έως button4_6) ακριβώς όσα και τα τετράγωνα του 4x6 χώρου τα οποία ωστόσο είναι μη επιλέξιμα (non-clickable). Τα πλήκτρα αυτά παραμένουν ενεργά για 30 δευτερόλεπτα ακριβώς για όσο πρέπει οι συμμετέχοντες να κωδικοποιήσουν τον χώρο.



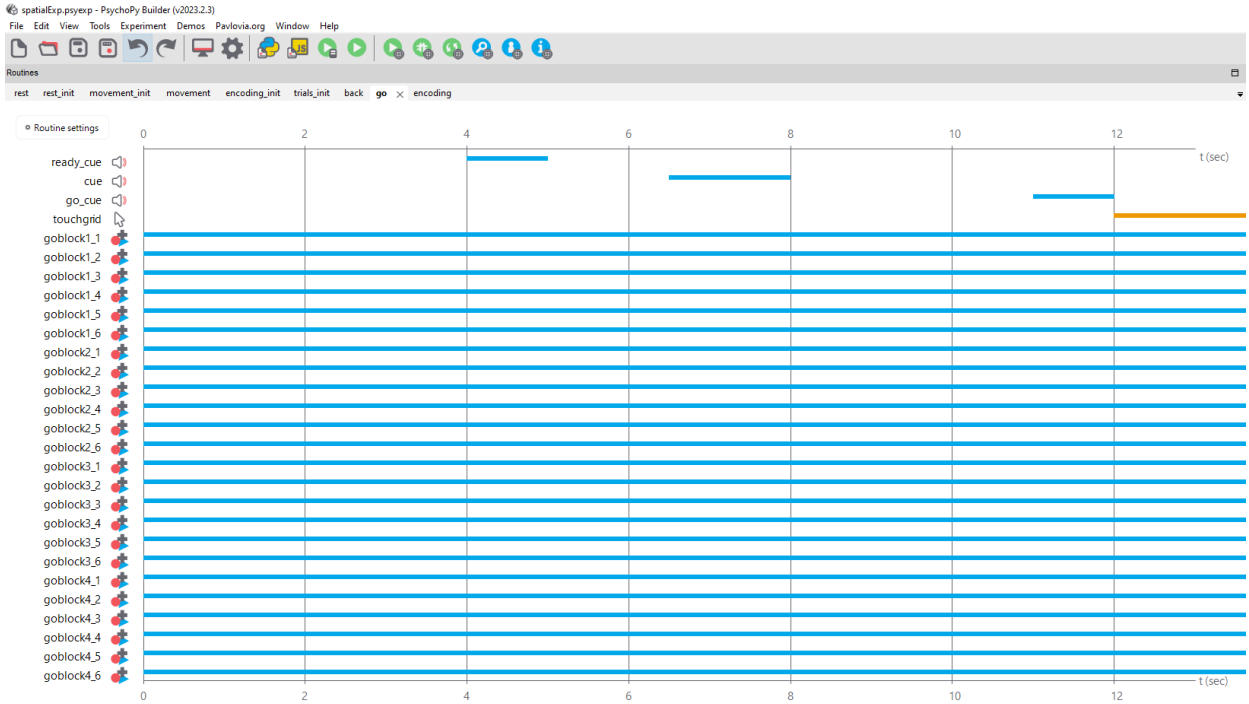
Εικόνα 27 Φάση 3: *Ο ήχος της έναρξης, η εμφάνιση όλων των 24 κελιών του πλέγματος στην οθόνη για 30 δευτερόλεπτα και ο ήχος της λήξης*

Κατά τη διάρκεια του τέταρτου και τελευταίου μέρους λαμβάνει χώρα το κύριο μέρος του πειράματος και οι δύο Ομάδες ακολούθησαν ακριβώς τις ίδιες οδηγίες. Το σύνολο των συμμετεχόντων είχε κλειστά τα μάτια με μία μάσκα. Στην αρχή τους ζητήθηκε να παραμείνουν σε ηρεμία μέχρι να ακούσουν τον χαρακτηριστικό ήχο. Στη συνέχεια, ακολουθούσε η ονομασία της θέσης του πλέγματος στο οποίο καλούνταν να οδηγήσουν τον δείκτη του κυρίαρχου χεριού τους, για παράδειγμα «δύο κόμμα τρία» σημαίνει δεύτερο τετράγωνο της τρίτης σειράς του πλέγματος. Μόλις ο χαρακτηριστικός ήχος ακουγόταν για δεύτερη φορά η κίνηση του χεριού έπρεπε να ξεκινήσει και ο συμμετέχοντας άγγιζε την επιθυμητή θέση. Για την αποφυγή πιθανών σφαλμάτων η επιθυμητή θέση έπρεπε να δηλωθεί με διπλή πίεση (double tap). Τέλος, όταν ο ήχος ξανακουγόταν, από τον συμμετέχοντα ζητούταν να επιστρέψει το χέρι του στην αρχική σταθερή θέση. Η αρχική θέση ορίζεται ως ένα σταθερό σημείο το οποίο ακουμπάει συνεχώς το αριστερό χέρι και το δεξί, το οποίο συμμετέχει στο πείραμα, ακουμπάει το αριστερό ώστε πάντα να ξεκινάει την κίνησή του από συγκεκριμένο σημείο.

Όταν ο επιτηρητής του πειράματος έβλεπε ότι ο συμμετέχοντας έχει όντως επιστρέψει το κυρίαρχο χέρι του στην προκαθορισμένη θέση έδινε την κατάλληλη εντολή στον υπολογιστή πατώντας το πλήκτρο ‘space’ και η επόμενη επανάληψη της τέταρτης φάσης ξεκινούσε με 2 δευτερόλεπτα κενό, για να ξεκουραστεί ο συμμετέχοντας, και με τα ίδια ακριβώς χαρακτηριστικά και ζητούμενα, εκτός

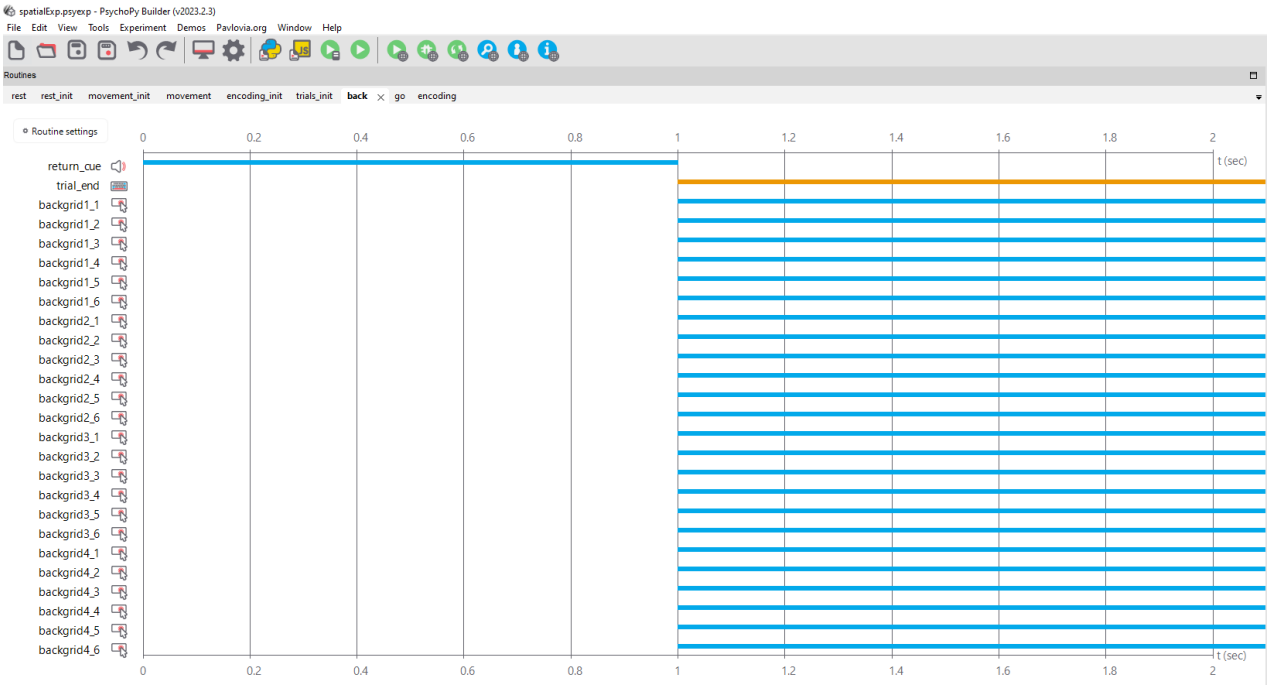
από την θέση η οποία ζητούταν αυτήν τη φορά από τον συμμετέχοντα. Συνολικά οι επαναλήψεις της τέταρτης φάσης ήταν 40.

Στις εικόνες που ακολουθούν φαίνεται η υλοποίηση της τέταρτης και τελευταίας φάσης του πειράματος. Στην αρχή των στοιχείων αυτήν τη φορά υπάρχουν τρία ηχητικά σήματα. Στην αρχή υπάρχει το ready_cue που προετοιμάζει τον συμμετέχοντα ότι η θέση του χώρου την οποία καλείται να αγγίξει θα ανακοινωθεί μέσω ηχητικού σήματος σύντομα, στην συνέχεια μετά από ακριβώς ένα δευτερόλεπτο ο ήχος cue παίζει. Ο ήχος cue είναι επίσης πάντα διάρκειας ενός δευτερολέπτου και αποκαλύπτει την θέση που ο συμμετέχοντας πρέπει να βρει. Μέσα σε αυτό το στοιχείο υπάρχουν φορτωμένα όλα τα πιθανά σημεία του χώρου ωστόσο το πρόγραμμα διαλέγει δυναμικά και τυχαία κάθε φορά ένα άλλο σημείο. Στην συνέχεια, μετά από παύση ενός δευτερολέπτου, το ηχητικό σήμα (go_cue) που σηματοδοτεί την λήξη του στοιχείου και την έναρξη της δοκιμής από τον συμμετέχοντα παίζει. Μετά, υπάρχουν πάλι 24 στοιχεία αφής αλλά αυτήν την φορά είναι προγραμματισμένα να αναγνωρίζουν την πίεση και όταν αντιληφθούν διπλή πίεση να τερματίσουν αυτό το στάδιο ώστε να περάσει το πείραμα στα επόμενα στοιχεία.



Εικόνα 28 Φάση 4: *Η ανακοίνωση της θέσης που πρέπει να πιέσει ο συμμετέχοντας και η πίεση*

Μετά την διπλή πίεση ξεκινάει η διαδικασία επιστροφής του συμμετέχοντα στην θέση αναφοράς ώστε να προετοιμαστεί για την επόμενη επανάληψη. Αυτό γίνετια μέσω του χαρακτηριστικού ήχου διάρκειας ενός δευτερολέπτου (return_cue). Όταν ο παρατηρητής του πειράματος σιγουρευτεί ότι το κυρίαρχο χέρι του συμμετέχοντα βρίσκεται και πάλι στο σημείο αναφοράς πατάει το πλήκτο space για τον τερματισμό, εντολή που εμπεριέχεται στο στοιχείο πληκτρολογίου (trial_end).

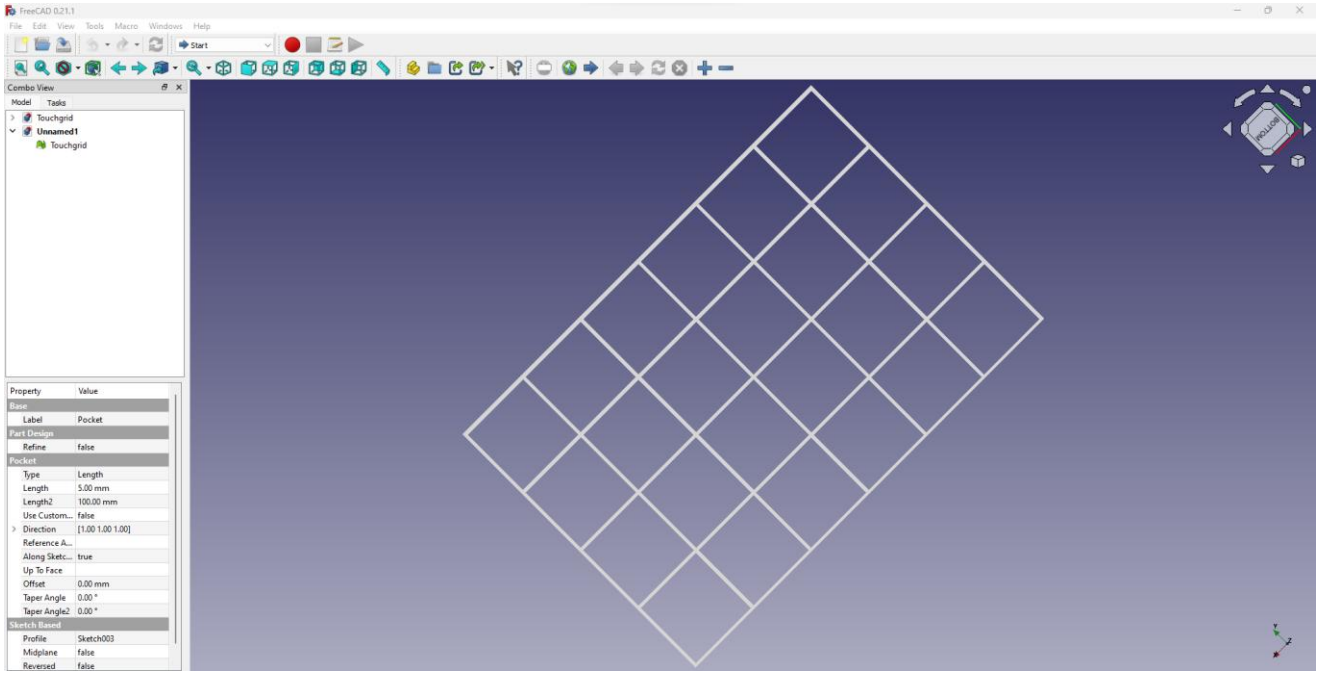


Εικόνα 29 Φάση 4: *Ο ήχος λήξης της επανάληψης και η πίεση του πλήκτρου 'space' από τον επιτηρητή*

Η συνολική χρονική έκταση του πειράματος κατά μέσο όρο από όλους τους συμμετέχοντας ήταν 850 δευτερόλεπτα (περίπου 14 λεπτά).

Ο χώρος που οι συμμετέχοντες σε αυτό το πείραμα κλήθηκαν να κωδικοποιήσουν και στην συνέχεια να προσανατολιστούν ήταν ένα πλέγμα 4x6 που βρισκόταν στη μία οθόνη αφής Pi Display 7 ιντσών και ανάλυσης 800x480 της κατασκευάστριας εταιρείας Waveshare, η οποία ήταν άμεσα συνδεδεμένη με τον υπολογιστή που ήλεγχε τη ροή του πειράματος.

Όπως προαναφέρθηκε με σκοπό να σταθεί δυνατή η ροή του πειράματος οι συμμετέχοντες που ήταν στην ομάδα παρεμποδισμένης όρασης έπρεπε να μπορούν με την βοήθεια της αφής να διακρίνουν το πλέγμα τετραγώνων όταν αυτό εμφανιζόταν πάνω στην οθόνη. Για τον λόγο αυτό, ένα ανάγλυφό πλέγμα που προσομοίωνε την εικόνα που εμφάνιζε η οθόνη κατασκευάστηκε με τη βοήθεια τρισδιάστατου εκτυπωτή. Το περί λόγου πλέγμα δημιουργήθηκε με τη βοήθεια του λογισμικού FreeCAD.



Εικόνα 30 *Κάτοψη από τον σχεδιασμό του πλέγματος*

Για την καταγραφή των διεργασιών του εγκεφάλου χρησιμοποιήθηκε το κράνος καταγραφής (cap) Ultracortex "Mark IV" της OpenBCI [77], [78], [79], ένα εκτυπώσιμο τριών διαστάσεων (3D printed) που επιτρέπει την λήψη σήματος μέχρι και 16 καναλιών. Τα κανάλια λήψης μπορούν να είναι οποιαδήποτε από τις 35 θέσεις του συστήματος 10-20 που αυτό το cap διαθέτει, δίνοντας έτσι της ευκαιρία μεγάλης προσαρμοστικότητας στο εκάστοτε πείραμα αφού μπορεί να γίνει επιλογή ανάμεσα στις 35 θέσεις με βάση τις ιδιαίτερες ανάγκες της κάθε περίστασης. Επιπλέον, αυτό το μοντέλο χρησιμοποιεί στεγνά (dry) ηλεκτρόδια, προσφέροντας την ικανότητα αδιάκοπης καταγραφής αφού δεν απαιτούνται τακτικές παύσεις για εφαρμογή γέλης στα σημεία. Η επιλογή του συγκεκριμένου τύπου ηλεκτροδίων διευκολύνει τόσο τον εξεταζόμενο, καθώς ο συνολικός χρόνος και η πολυπλοκότητα εγκατάστασης μειώνονται, αλλά και το ίδρυμα αφού μειώνεται το οικονομικό κόστος [80], [81], [82], [83], [84], [85].

Η OpenBCI διαθέτει, επίσης, το σύμπλεγμα πλακετών Cyton και Daisy που δίνουν την δυνατότητα λήψης σήματος μέχρι 16 καναλιών με επεξεργαστή 32-bit και μπορούν να αντιστοιχισθούν σε ισάριθμα κανάλια EEG. Η πλακέτα Cyton είναι η βασική των δύο πλακετών και πραγματοποιεί την σύνδεση με τα ηλεκτρόδια, την μετατροπή του σήματος σε ψηφιακό και την αποστολή των δεδομένων μέσω Bluetooth. Η μεταφορά μέσω Bluetooth πραγματοποιείται με την βοήθεια και ενός προσαρμογέα (dongle) που παρέχεται από την OpenBCI, ο οποίος λαμβάνει τα δεδομένα και με τη σειρά του τα μεταφέρει στον υπολογιστή. Η δευτερεύουσα πλακέτα του συμπλέγματος, η Daisy, είναι επί της ουσίας μία επέκταση του προαναφερθέντος συστήματος προσθέτοντας την δυνατότητα υποστήριξης επιπλέον 8 καναλιών, φτάνοντας τελικά των συνολικό αριθμό στα 16. [86], [87], [88]

Για να σταθεί δυνατή η παραπάνω πειραματική διάταξη έπρεπε να εκτυπωθούν σε 3D εκτυπωτή πλήθος εξαρτημάτων. Η εκτύπωση και κατασκευή των κάτωθι δεν έγινε στα πλαίσια της συγκεκριμένης εργασίας. Ωστόσο θα γίνει μία σύντομη αναφορά στην διαδικασία που ακολουθήθηκε.

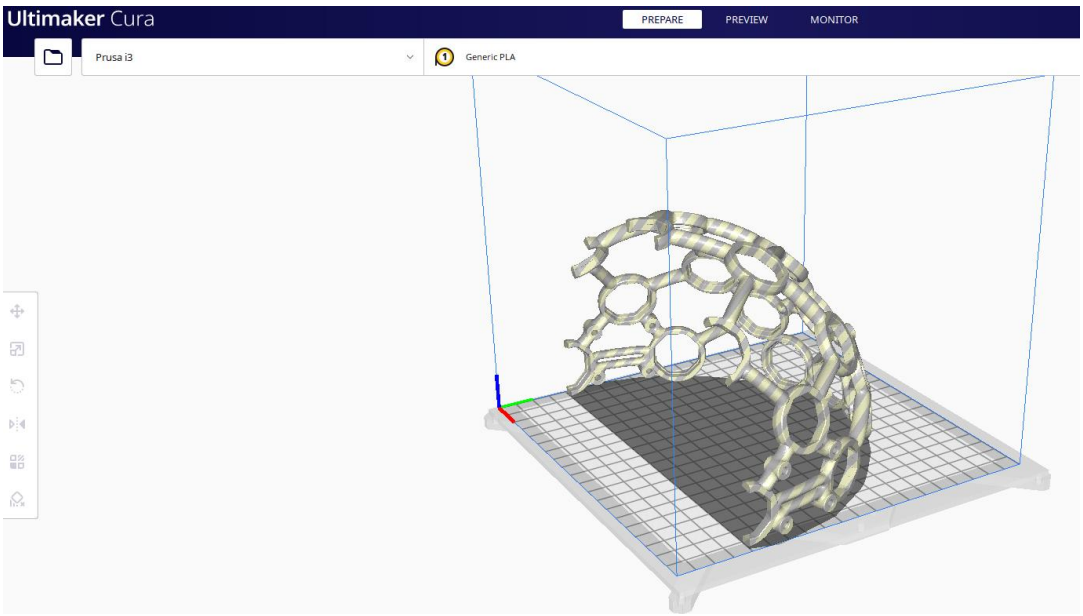
Έπρεπε να εκτυπωθούν τα εξαρτήματα:

- Σκελετός κάσκας καταγραφής (frame)
- Εξαρτήματα σκελετού για την προσαρμογή των ηλεκτροδίων (mech parts)
- Εξαρτήματα σκελετού για την προσαρμογή των καλωδίων (wire parts)

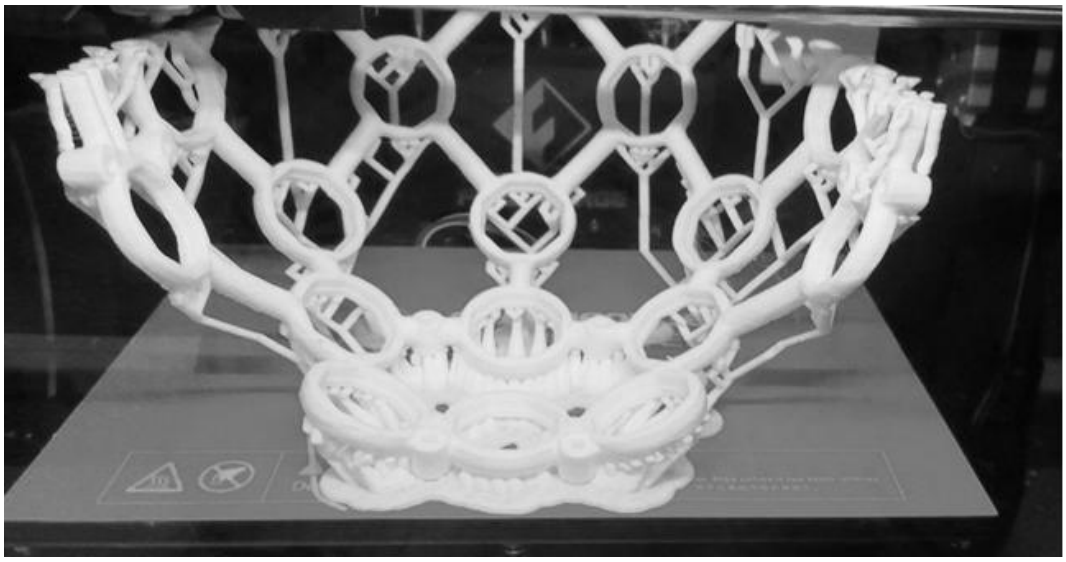
- Θήκη για την τοποθέτηση του συμπλέγματος πλακετών Cyton+Daisy πάνω στην κάσκα καταγραφής (board mount και board cover)

Για τα παραπάνω χρησιμοποιήθηκαν τα αρχεία που παραχωρήθηκαν από την OpenBCI, τα οποία προσαρμόστηκαν στον τρισδιάστατο εκτυπωτή με την βοήθεια των πακέτων λογισμικού PursaSlicer (λογισμικό της εταιρείας στην οποία υπάγεται ο χρησιμοποιούμενος εκτυπωτής) και Ultimaker Cura (πρόγραμμα γενικού σκοπού για τρισδιάστατες εκτυπώσεις).

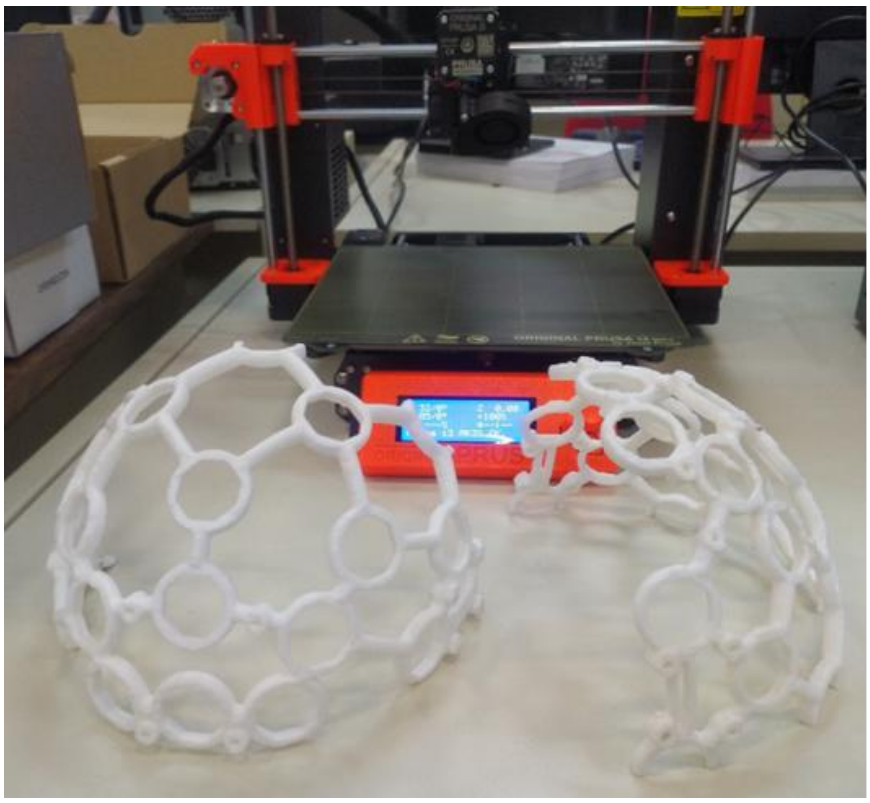
Ο σκελετός του ηλεκτροεγκεφαλογράφου εκτυπώθηκε σε δύο διαφορετικά κομμάτια σε ισάριθμες διακριτές διαδικασίες.



Εικόνα 31 *Στιγμιότυπο από το λογισμικό PrusaSlicer για την εκτύπωση του σκελετού*

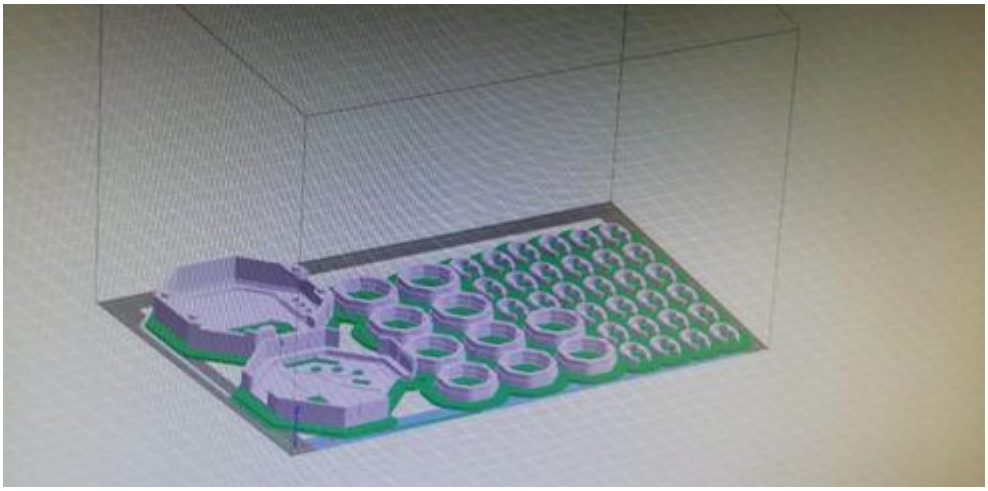


Εικόνα 32 *Εκτύπωση του σκελετού*

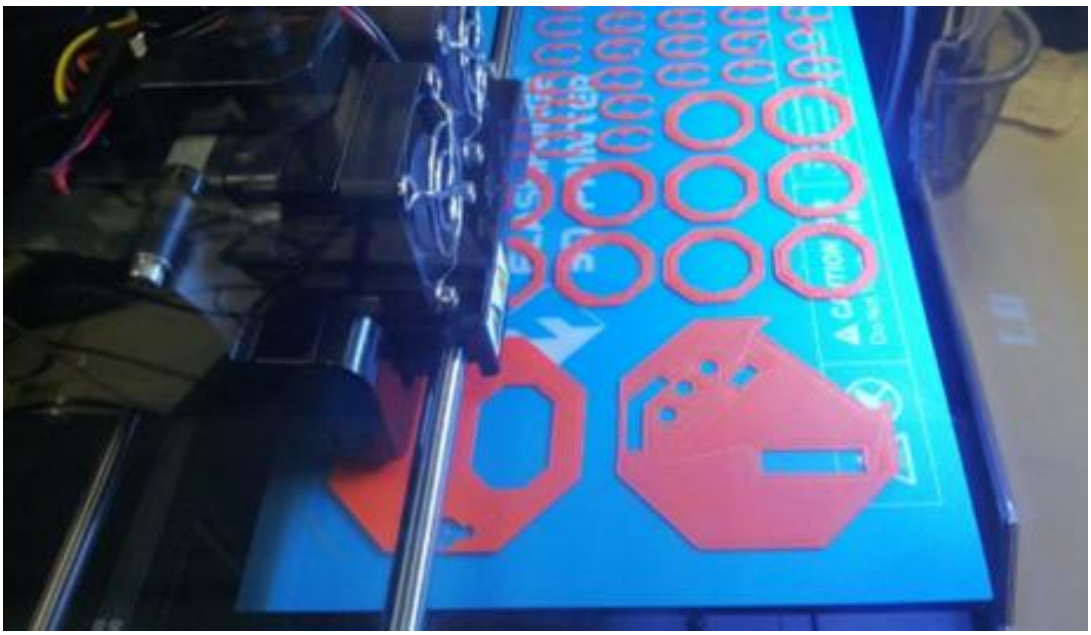


Εικόνα 33 *Τα 2 διακριτά κομμάτια του σκελετού ηλεκτροεγκεφαλογράφου εκτυπωμένα*

Το σύνολο των υπόλοιπων εξαρτημάτων (mech parts, wire clips, board mount, board cover) έγινε σε κοινό αρχείο και διαδικασία η οποία φαίνεται παρακάτω.

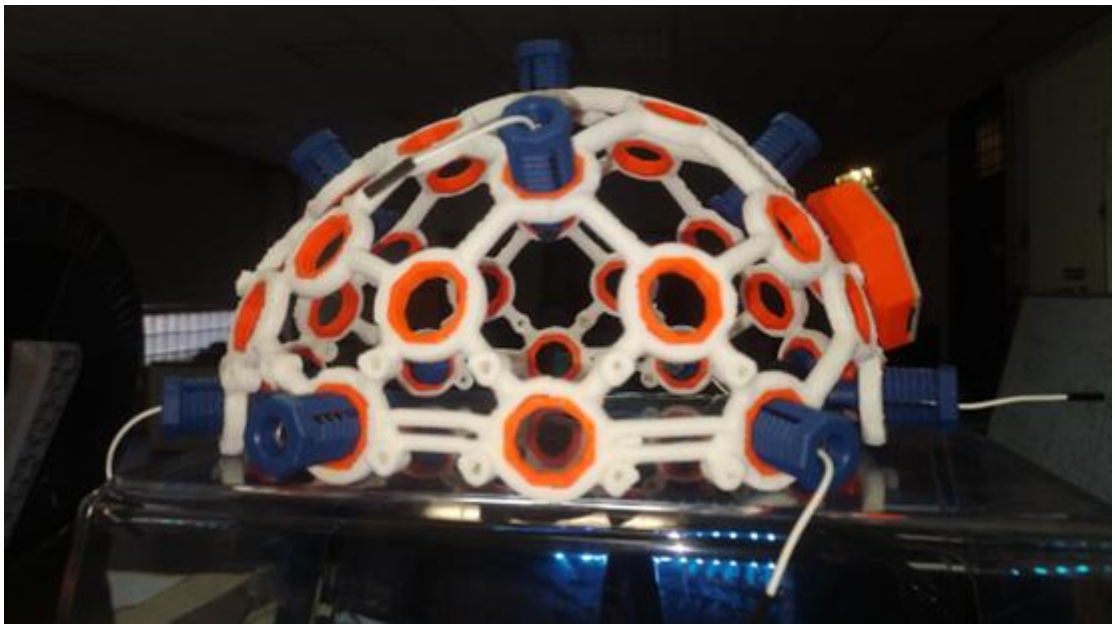


Εικόνα 34 *Σχεδίαση αρχείου τρισδιάστατης εκτύπωσης με τα υπόλοιπα εξαρτήματα της κάσκας*

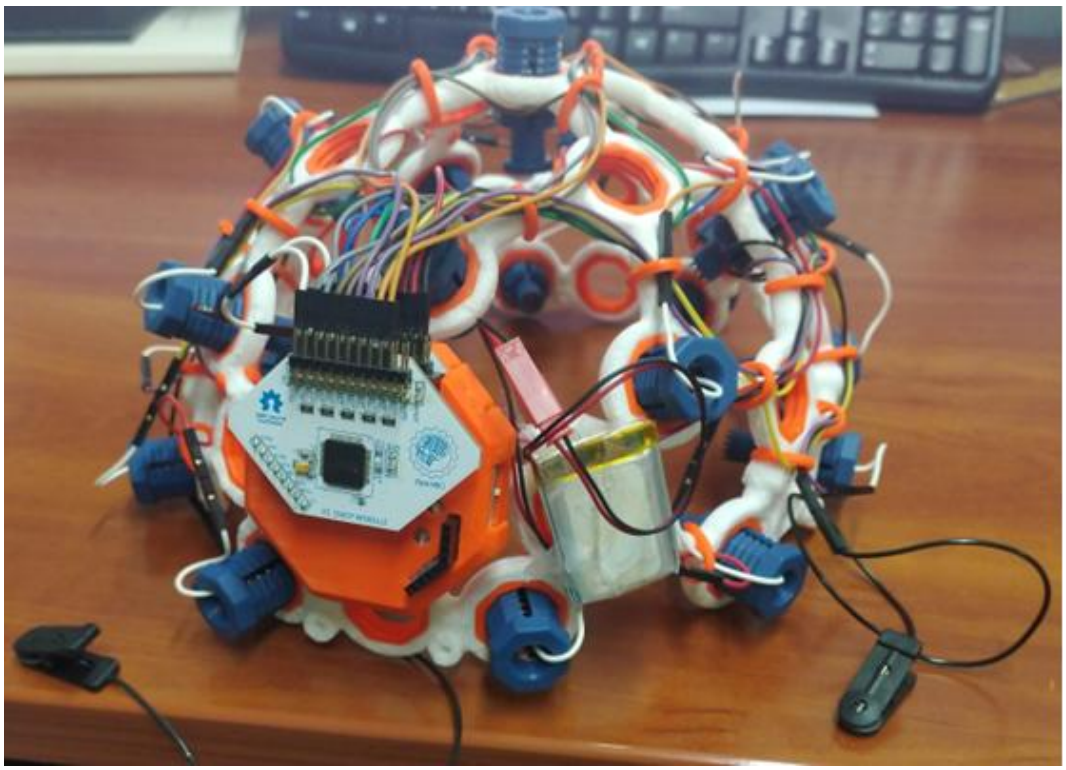


Εικόνα 35 *Εκτύπωση των εξαρτημάτων - Αρχικό στάδιο*

Τα επιμέρους εξαρτήματα προσαρμόστηκαν στον σκελετό του ηλεκτροεγκεφαλογράφου, και μετά την εφαρμογή όλων των εξαρτημάτων, των ηλεκτροδίων και των πλακετών Cyton+Daisy μέσα στην κατάλληλη θήκη προέκυψε το τελικό αποτέλεσμα που φαίνεται στην συνέχεια:



Εικόνα 36 *Ολοκληρωμένη κάσκα ηλεκτροεγκεφαλογράφου OpenBCI*



Εικόνα 37 *Σύνδεση ηλεκτροδίων με το σύστημα Cyton+Daisy*

Όσον αφορά την σύνδεση των καλωδίων στις πλακέτες Cyton+Daisy, επισημαίνεται ότι κατά την ανάπτυξη του λογισμικού και τη δημιουργία του διαύλου καταγραφής ηλεκτροεγκεφαλογραφήματος, δόθηκε προσοχή ώστε οι ετικέτες των καναλιών καταγραφής που ορίστηκαν στο λογισμικό που αναπτύχθηκε να ταυτίζονται με τις συνδέσεις των καναλιών σε επίπεδο υλικού σύμφωνα με τον παρακάτω πίνακα:

Πίνακας 2 *Ηλεκτρόδια ηλεκτροεγκεφαλογράφου OpenBCI*

Electrode	Board	Pin
Ear Clip	Cyton	Bottom SRB pin
FP1	Cyton	Bottom N1P pin
FP2	Cyton	Bottom N2P pin
C3	Cyton	Bottom N3P pin
C4	Cyton	Bottom N4P pin
P7	Cyton	Bottom N5P pin
P8	Cyton	Bottom N6P pin
O1	Cyton	Bottom N7P pin
O2	Cyton	Bottom N8P pin
Ear Clip	Cyton	Bottom BIAS pin
F7	Daisy	Bottom N1P pin
F8	Daisy	Bottom N2P pin
F3	Daisy	Bottom N3P pin
F4	Daisy	Bottom N4P pin
T7	Daisy	Bottom N5P pin
T8	Daisy	Bottom N6P pin
P3	Daisy	Bottom N7P pin
P4	Daisy	Bottom N8P pin

7.4. Λήψη Δεδομένων

Για την εξαγωγή των πειραματικών αποτελεσμάτων πραγματοποιήθηκαν καταγραφές ηλεκτροεγκεφαλογραφήματος (ΗΕΓ) σε συνολικά δέκα (10) υγιείς εθελοντές, ηλικίας 25 έως 35 ετών, εκ των οποίων έξι (6) ήταν άνδρες και τέσσερις (4) γυναίκες. Οι συμμετέχοντες δεν είχαν ιστορικό νευρολογικών ή ψυχολογικών διαταραχών και έδωσαν προφορική συγκατάθεση για τη συμμετοχή τους στο πείραμα. Το πείραμα πραγματοποιήθηκε σε εργαστήριο του Εθνικού Μετσοβίου Πολυτεχνείου, σε ελεγχόμενο περιβάλλον με περιορισμένες εξωτερικές παρεμβολές. Οι συμμετέχοντες χωρίστηκαν σε δύο ισάριθμες ομάδες σύμφωνα με το πειραματικό πρωτόκολλο. Η πρώτη ομάδα (Control Group) κωδικοποίησε τον χώρο με τη χρήση της όρασης και εκτέλεσε το κυρίως μέρος του πειράματος με παρεμποδισμένη όραση. Η δεύτερη ομάδα (Study Group) εκτέλεσε το πείραμα εξ ολοκλήρου χωρίς οπτικά ερεθίσματα, με καλυμμένα μάτια καθ' όλη τη διάρκεια. Και στις δύο περιπτώσεις, οι συμμετέχοντες έπρεπε να εκτελέσουν τις ίδιες φάσεις του πειράματος, όπως αυτές είχαν αναπτυχθεί στο λογισμικό PsychoPy, ακολουθώντας τις ηχητικές οδηγίες και εκτελώντας τις αντίστοιχες χωρικές εργασίες.

Η προετοιμασία κάθε συμμετέχοντα περιελάμβανε αρχικά σύντομη ενημέρωση και επεξήγηση των στόχων του πειράματος, καθώς και αναλυτική παρουσίαση των επιμέρους φάσεών του. Στη συνέχεια, πραγματοποιήθηκε η τοποθέτηση της κάσκας ηλεκτροεγκεφαλογραφήματος OpenBCI Ultracortex Mark IV, με ταυτόχρονη προσπάθεια βελτιστοποίησης της επαφής των ηλεκτροδίων με το τριχωτό της κεφαλής. Για τον σκοπό αυτό χρησιμοποιήθηκε η διεπαφή χρήστη (GUI) της OpenBCI, η οποία παρέχει σε πραγματικό χρόνο ενδείξεις για την αντίσταση επαφής κάθε ηλεκτροδίου. Μετά την ορθή τοποθέτηση και τον έλεγχο της κάσκας, κάθε συμμετέχων λάμβανε οδηγίες για να παραμείνει άνετος και ακίνητος καθ' όλη τη διάρκεια του πειράματος. Η συνολική διάρκεια του πειράματος για κάθε συμμετέχοντα ήταν περίπου δεκατέσσερα (14) λεπτά, μαζί με επιπλέον 25 λεπτά περίπου που απαιτούνταν για την επεξήγηση και την τοποθέτηση των ηλεκτροδίων. Η διαχείριση της παροχής των ερεθισμάτων και της καταγραφής των events πραγματοποιήθηκε μέσω του λογισμικού PsychoPy.

Για κάθε συμμετέχοντα δημιουργήθηκαν αυτόματα δύο αρχεία δεδομένων σε μορφή .csv. Το πρώτο αρχείο περιείχε την πλήρη καταγραφή του ηλεκτροεγκεφαλογραφήματος, ενώ το δεύτερο περιείχε όλα τα events του πειράματος, όπως την εμφάνιση των ερεθισμάτων, την έναρξη και λήξη κάθε φάσης και τις χρονικές στιγμές απόκρισης του συμμετέχοντα, σύμφωνα με τον κώδικα του PsychoPy που παρατίθεται στο Παράρτημα Α. Όλα τα δεδομένα αποθηκεύτηκαν με ανώνυμο τρόπο, χρησιμοποιώντας μοναδικό κωδικό αναγνώρισης για κάθε συμμετέχοντα, ώστε να διασφαλιστεί η προστασία των προσωπικών δεδομένων.

8. Ανάλυση Δεδομένων

Για την ανάλυση των δεδομένων του ηλεκτροεγκεφαλογραφήματος που συλλέχθηκαν στα πλαίσια αυτού του πειράματος χρησιμοποιήθηκε το MATLAB και συγκεκριμένα η έκδοση R2022b, με τον αναλυτικό κώδικα να περιλαμβάνεται στο Παράρτημα Β. Το MATLAB είναι ένα περιβάλλον αριθμητικής υπολογιστικής και μια γλώσσα προγραμματισμού πολλαπλών παραδειγμάτων που αναπτύχθηκε από την εταιρεία MathWorks και περιλαμβάνει μεγάλο πλήθος πακέτων ανάλυσης δεδομένων. Ένα από αυτά είναι το EEGLAB που χρησιμοποιήθηκε εκτενώς για αυτήν την ανάλυση. Το EEGLAB είναι ένα διαδραστικό πακέτο (toolbox) για την επεξεργασία συνεχών και σχετιζόμενων με συμβάντα (event-related) EEG, MEG και άλλα ηλεκτροφυσιολογικών δεδομένων ενσωματώνοντας ανάλυση ανεξάρτητων συνιστωσών (ICA), ανάλυση χρόνου/συχνότητας, απόρριψη τεχνικών σφαλμάτων και πολλά άλλα χρήσιμα εργαλεία. Το EEGLAB διαθέτει επίσης μία γραφική διεπαφή (GUI) για πιο εύκολη επεξεργασία των δεδομένων, ωστόσο σε αυτήν την ανάλυση δεν χρησιμοποιήθηκε. Αντ' αυτού, υπάρχει η εντολή 'eeglab nogui;' έτσι ώστε να εκκινηθεί το EEGLAB χωρίς όμως τη γραφική διεπαφή. Επιπλέον, το EEGLAB διαβάζει τα παραγόμενα αρχεία από γνωστά μοντέλα ηλεκτροεγκεφαλογράφων και τα μετατρέπουν αυτόματα στη μορφή που το πακέτο χρειάζεται για την ανάλυση. Ωστόσο δυστυχώς δεν υπάρχει τέτοια συνάρτηση να διαβάζει .csv αρχεία που παράγει ο ηλεκτροεγκεφαλογράφος που χρησιμοποιήθηκε στο παρόν πείραμα. Για να προσπεραστεί αυτή η δυσκολία, δόθηκε σαν είσοδος στο πρόγραμμα ένα ενδεικτικό αρχείο ΗΕΓ τύπου .bdf (από διαφορετικό πείραμα), το οποίο διαβάστηκε από την συνάρτηση pop_biosig και εξήγαγε τα δεδομένα στην κατάλληλη μορφή (struct eeg_demo). Στη συνέχεια, ένα .csv αρχείο με δεδομένα από το παρόν πείραμα εισήχθη στο MATLAB με τη συνάρτηση readtable και αποθηκεύτηκε σε ξεχωριστή μεταβλητή exp_data, προκειμένου να ενσωματωθούν στη δομή eeg_demo, η οποία μπορεί να χρησιμοποιηθεί από το EEGLAB.

Για τον σκοπό αυτό, αρχικά ορίζονται οι βασικές μεταβλητές του πειράματος στη δομή eeg_demo όπως ο αριθμός των καναλιών και ο ρυθμός καταγραφής (eeg_channelnum = 16; eeg_sample_rate = 125;). Επιπλέον, ορίστηκε το πεδίο της δομής δεδομένων που αντιστοιχεί στα δεδομένα των καταγραφών, το οποίο αντιστοιχεί σε έναν πίνακα με στήλες όλα τα ηλεκτρόδια και γραμμές τις χρονικές στιγμές της καταγραφής για κάθε ηλεκτρόδιο. Επομένως, από τα δεδομένα της καταγραφής χρησιμοποιούνται 16 στήλες που αντιστοιχούν στον αριθμό των ηλεκτροδίων, ενώ απορρίπτονται στήλες που δεν αξιοποιούνται στο παρόν πείραμα. Αυτές διαγράφονται με την εντολή exp_data=removevars(exp_data,{'AccelChannel0', 'AccelChannel1', 'AccelChannel2', 'Other', 'Other_1', 'Other_2', 'Other_3', 'Other_4',' Other_5', 'Other_6', 'AnalogChannel0', 'AnalogChannel1', 'AnalogChannel2'}).

Έπειτα οι ονομασίες των ηλεκτροδίων της δομής eeg_demo μετονομάζονται ώστε να αντιστοιχούν σε αυτές του exp_data και του μοντέλου 10-20 που ακολουθείται. Οι τοπογραφικές πληροφορίες για το πού βρίσκεται το κάθε ηλεκτρόδιο πάνω στο κρανίο εισάγονται από ένα εξωτερικό αρχείο ονόματος chanlogs και εισάγονται στην μεταβλητή της δομής eeg_demo.chanlogs.

Επιπρόσθετα, ορίστηκε το κέρδος (gain) σε 24, έτσι ώστε στη συνέχεια να υπολογιστεί ο συντελεστής κλίμακας που πρέπει να εφαρμοστεί (με πολλαπλασιασμό) στις μετρήσεις της συγκεκριμένης διάταξης όπως αυτές εμφανίζονται στα ακατέργαστα δεδομένα, ώστε να προκύψουν οι πραγματικές μετρούμενες τιμές.

$$eeg\ scale\ factor = {}^{4.5}/_{gain}/_{(2^{23}-1)}\ {}^{Volts}/_{count}$$

Μετά την ολοκλήρωση της παραπάνω διαδικασίας, προκύπτει η τελική μορφή της δομής eeg_exp που περιλαμβάνει όλες τις πληροφορίες του πειράματος στη σωστή μορφή για περαιτέρω επεξεργασία. Συνοπτικά, οι βασικές παράμετροι που ορίστηκαν περιλαμβάνουν τις εξής: setname, filename, nbchan (συνολικός αριθμός ηλεκτροδίων), srate (ρυθμός δειγματοληψίας), xmin (ελάχιστη τιμή των δεδομένων), xmax (μέγιστη τιμή των δεδομένων), times (η γραμμική χρονική ακολουθία των στιγμών κατά των οποίων οι τιμές των ηλεκτροδίων καταγράφονταν), data (οι μετρήσεις που κατέγραφαν τα ηλεκτρόδια), pnts (ο αριθμός των καρέ/χρονικών σημείων ανά πειραματική εποχή).

Αρχικά χρησιμοποιείται η συνάρτηση pop_reref, συνάρτηση που παρέχει το πακέτο του EEGLAB, η οποία υπολογίζει τον μέσο όρο του σήματος από όλα τα ηλεκτρόδια του ΗΕΓ και τον αφαιρεί από κάθε ηλεκτρόδια για κάθε χρονική στιγμή, μέθοδος γνωστή και ως average reference. Στη συνέχεια, η έξοδος της συνάρτησης αυτής περνάει ως είσοδος στη συνάρτηση pop_resample η οποία επαναδειγματοληπτεί το σήμα. Ο ρυθμός δειγματοληψίας μετράται σε Hertz (Hz) και καθορίζει τον ρυθμό, ανά δευτερόλεπτο, με τον οποίο έγινε η δειγματοληψία του αρχικού αναλογικού σήματος. Όσο υψηλότερος είναι ο ρυθμός δειγματοληψίας, τόσο μεγαλύτερη είναι η χρονική ακρίβεια της αναπαράστασης του σήματος. Εδώ το σήμα επαναδειγματοληπτείται στα 125Hz.

Μελέτες και μετρήσεις έχουν δείξει ότι όλη η χρήσιμη πληροφορία σε ένα ΗΕΓ βρίσκεται μεταξύ στα 1-50Hz. Αυτός είναι ο λόγος που στη συνέχεια η δομή περνάει από ένα ζωνοπερατό φίλτρο που κόβει τις συχνότητες χαμηλότερα από 1Hz και αυτές ψηλότερα από 50 Hz.

Έπειτα, τα δεδομένα της δομής eeg_data περνάνε από τη συνάρτηση detrend για αφαίρεση τυχόν γραμμικής συνιστώσας. Η συνάρτηση pop_runica είναι η επόμενη στην οποία προωθούνται σαν είσοδο τα δεδομένα από την έξοδο της detrend. Αυτή η συνάρτηση τρέχει την ανάλυση των ανεξαρτήτων συνιστωσών (ICA). Με την βοήθεια αυτής της ανάλυσης μπορούν να αφαιρεθούν τυχόν θόρυβοι που μπορεί να είναι ενσωματωμένοι στα δεδομένα όπως για παράδειγμα κίνηση μυών, ανοιγοκλείσιμο ματιών ή κίνηση αυτών. Τα αποτελέσματα της ICA, δηλαδή οι ανεξάρτητες συνιστώσες που εντοπίστηκαν απεικονίζονται με την βοήθεια της συνάρτησης pop_topoplot σε διάγραμμα. Το EEGLAB παρέχει την δυνατότητα μέσω της iclabel αυτόματης ταξινόμησης των συνιστωσών, όπου μπορεί να αποφανθεί με ποσοστά τί μπορεί να αντιπροσωπεύει η κάθε συνιστώσα (εγκεφαλική δραστηριότητα, κίνηση ματιών, κίνηση μυών, κ.ά.). Με τη βοήθεια της παραπάνω ταξινόμησης η συνάρτηση pop_subcomp μπορεί να αφαιρέσει από το σύνολο των δεδομένων την συνιστώσα που αποτελεί θόρυβο για τις εκάστοτε ανάγκες, φτάνει να θέσει κανείς σαν παράμετρο στην συνάρτηση της συνιστώσα που θέλει να αφαιρέσει. Μετά την εκτέλεση της ανάλυσης ανεξάρτητων συνιστωσών, εφαρμόστηκε η διαδικασία αυτόματης ταξινόμησης μέσω του εργαλείου ICLabel του EEGLAB. Το ICLabel υπολογίζει την πιθανότητα κάθε συνιστώσας να αντιπροσωπεύει εγκεφαλική δραστηριότητα ή πηγή θορύβου, όπως κινήσεις ματιών, μυϊκές συσπάσεις ή ηλεκτρικές παρεμβολές. Με βάση τα αποτελέσματα της ταξινόμησης, απορρίφθηκαν επιλεκτικά οι συνιστώσες που εμφάνιζαν υψηλή συσχέτιση με μη εγκεφαλικές πηγές, χρησιμοποιώντας τη συνάρτηση pop_subcomp. Με αυτόν τον τρόπο διατηρήθηκαν μόνο οι καθαρές εγκεφαλικές συνιστώσες, βελτιώνοντας σημαντικά την ποιότητα του σήματος για τα επόμενα στάδια ανάλυσης. Με την αφαίρεση και των συνιστωσών που αποδίδονται σε θόρυβο, η επεξεργασία του σήματος ΗΕΓ που λαμβάνεται από τους συμμετέχοντες ολοκληρώνεται.

Στη συνέχεια πραγματοποιήθηκε ανάλυση στο πεδίο της συχνότητας μέσω Μετασχηματισμού Fourier (FFT), προκειμένου να υπολογιστεί η ενεργειακή κατανομή των σημάτων στις βασικές ζώνες συχνοτήτων. Μετά τον μετασχηματισμό εφαρμόστηκαν ζωνοπερατά φίλτρα (bandpass) για την απομόνωση επιμέρους ζωνών: δ (1-4 Hz), θ (4-8 Hz), άλφα (8-13 Hz), β (13-30 Hz) και γ (30-50 Hz). Η απομόνωση αυτών των φασματικών περιοχών επέτρεψε τη σύγκριση της φασματικής ισχύος μεταξύ

των δύο πειραματικών ομάδων και την εξαγωγή ενδείξεων για πιθανές διαφοροποιήσεις στη νευρωνική δραστηριότητα υπό τις διαφορετικές συνθήκες όρασης και απουσίας όρασης.

Μετά το πέρας της ανάλυσης όλων των δεδομένων που συλλέχθηκαν από όλα τα άτομα που συμμετείχαν στο πείραμα, αναζητήθηκαν στατιστικά σημαντικές διαφορές στα δεδομένα του ηλεκτροεγκεφαλογραφήματος ανάμεσα σε Ομάδα 1 και Ομάδα 2 κατά τη διεξαγωγή του πειράματος. Το συμπέρασμα αυτό θα εξαχθεί με τη βοήθεια της στατιστικής ανάλυσης, όπου αναζητείται με διάφορες μεθόδους ο προσδιορισμός του βαθμού εμπιστοσύνης στην εξαγωγή ασφαλών συμπερασμάτων μέσα από περιορισμένο δείγμα δεδομένων ενός ευρύτερου συνόλου. Σημαντικό εργαλείο της στατιστικής ανάλυσης που βοηθάει τον παραπάνω σκοπό είναι τα στατιστικά τεστ (statistical hypothesis tests).

Τα στατιστικά τεστ είναι μαθηματικά εργαλεία για την ανάλυση ποσοτικών δεδομένων που προκύπτουν από μία ερευνητική μελέτη. Κάθε ερευνητής έχει έναν μεγάλο αριθμό στατιστικών τεστ στη φαρέτρα του από τα οποία το καθένα πρέπει, όμως, να χρησιμοποιείται κάτω από συγκεκριμένες συνθήκες και αλλάζει αναλόγως με το επιστημονικό ερώτημα που πρέπει να απαντηθεί, τη δομή των δεδομένων και το σχεδιασμό της μελέτης. Πριν καν την έναρξη της πειραματικής διαδικασίας και της επιλογής του στατιστικού τεστ πρέπει να διατυπωθεί η ερώτηση που πρέπει να απαντηθεί με αυτό το πείραμα καθώς και η μηδενική υπόθεση. Η μηδενική υπόθεση (null hypothesis) είναι ένα είδος υπόθεσης που θεωρείται ορθή μέχρι να αποδειχθεί το αντίθετο βάση των πειραματικών δεδομένων.

Αφού τα αρχικά ερωτήματα έχουν τεθεί και οι αρχικές υποθέσεις έχουν σχηματιστεί τα στατιστικά τεστ επιλέγονται λαμβάνοντας υπόψη τρία καίρια κριτήρια: τον αριθμό των μεταβλητών, τον τύπο των δεδομένων και το πρωτόκολλο σχεδιασμού του πειράματος [89].

Τα στατιστικά τεστ και οι διαδικασίες μπορούν να χωριστούν με βάση των αριθμό των μεταβλητών που είναι σχεδιασμένα να αναλύουν. Χωρίζονται σε τρείς ομάδες, αυτά που χρησιμοποιούνται σε απλές μεταβλητές, αυτά που χρησιμοποιούνται για να αναλύσουν την σχέση μεταξύ δύο μεταβλητών και τέλος αυτά που χρησιμοποιούνται για να μοντελοποιήσουν πολύ-μεταβλητές σχέσεις μεταξύ τριών ή περισσοτέρων μεταβλητών [89].

Μία άλλη κατηγοριοποίηση είναι ο τύπος των δεδομένων. Αναλόγως με τον σχεδιασμό του κάθε πειράματος και τον τρόπο που λαμβάνονται και κατηγοριοποιούνται τα δεδομένα, αυτά μπορεί να είναι συνεχή, κατηγοριοποιημένα ή δυαδικά [89].

Βάσει του πειραματικού σχεδιασμού, τα στατιστικά τεστ χωρίζονται σε δύο βασικές κατηγορίες: paired και unpaired. Στον paired σχεδιασμό, κάθε συμμετέχων υποβάλλεται σε όλες τις πειραματικές συνθήκες, επιτρέποντας έτσι τη σύγκριση μεταξύ διαφορετικών μετρήσεων του ίδιου ατόμου ή την αντιστοίχιση συμμετεχόντων μεταξύ των ομάδων με βάση κοινά χαρακτηριστικά. Για παράδειγμα, μπορεί να συγκριθούν δύο μετρήσεις από το ίδιο άτομο πριν και μετά από κάποια παρέμβαση ή να δημιουργηθούν ζεύγη ατόμων με παρόμοια χαρακτηριστικά για να ενταχθούν σε διαφορετικές ομάδες. Αντίθετα, στον unpaired σχεδιασμό κάθε συμμετέχων συμμετέχει μόνο σε μία πειραματική συνθήκη και συνεπώς οι μετρήσεις προέρχονται από ανεξάρτητες ομάδες, οι οποίες μπορεί να διαφέρουν και ως προς το πλήθος των ατόμων που περιλαμβάνουν. [89].

Το πείραμα που σχεδιάστηκε και πραγματοποιήθηκε για την συγκεκριμένη διπλωματική εργασία διαθέτει περισσότερες από δύο μεταβλητές, συνεχή δεδομένα και paired, αφού κάθε συμμετέχοντας διεξήγαγε μία φορά το πείραμα με ή χωρίς παρεμπόδιση της όρασής του.

Λαμβάνοντας υπόψη τα παραπάνω τα δύο στατιστικά τεστ που επιλέχθηκαν να χρησιμοποιηθούν είναι το Student’s t-test και το Wilcoxon’s rank sum test.

Το Student’s t-test είναι ένα τεστ για συνεχή δεδομένα που διερευνά αν οι μέσοι όροι για δύο πληθυσμούς είναι σημαντικά διαφορετικοί, υποθέτοντας ότι τα δεδομένα ακολουθούν την κανονική κατανομή. Δημοσιεύθηκε για πρώτη φορά το 1908 στο επιστημονικό περιοδικό Biometrika από τον William Sealy Gosset, χρησιμοποιώντας το ψευδώνυμο Student [90]. Το Wilcoxon’s rank sum test γνωστό επίσης και ως Mann-Whitney U test χρησιμοποιείται για συνεχή δεδομένα αλλά σε αντίθεση με το Student’s t-test δεν απαιτεί να είναι κανονικά κατανεμημένα [89].

9. Αποτελέσματα

Μετά την ολοκλήρωση της προεπεξεργασίας των σημάτων ΗΕΓ, πραγματοποιήθηκε η εξαγωγή των φασματικών χαρακτηριστικών και η στατιστική σύγκριση μεταξύ των δύο πειραματικών ομάδων. Για κάθε συμμετέχοντα υπολογίστηκε η μέση ισχύς ανά ζώνη και ανά ηλεκτρόδιο, καθώς και οι μέσοι όροι ανά ομάδα. Στον παρακάτω πίνακα παρουσιάζονται οι μέσες τιμές φασματικής ισχύος για αντιπροσωπευτικές περιοχές ενδιαφέροντος ανά ζώνη, συμπεριλαμβάνοντας στατιστικά σημαντικά αποτελέσματα ($p<0.1$) καθώς και πρόσθετα ηλεκτρόδια που θεωρούνται πιο σχετικά με τις αντίστοιχες ζώνες συχνοτήτων σύμφωνα με τη φυσιολογία.

Πίνακας 3 *Μέση φασματική ισχύς ανά ζώνη συχνοτήτων και πειραματική ομάδα*

Ζώνη Συχνοτήτων	**Περιοχή Εγκεφάλου**	**Ομάδα 1 (Παρεμπόδιση Όρασης)**	**Ομάδα 2 (Χωρίς Παρεμπόδιση)**	**Διαφορά (%)**	**p-value**	**Cohen's d (95% CI)**
Δέλτα (1-4 Hz)	Προμετωπιαία (Fp1–Fp2)	1.22 ± 0.26	1.18 ± 0.23	+3.4%	0.294	0.21 (-0.92 – 1.28)
θήτα (4-8 Hz)	Μετωπιαία (F3-F4)	6.85 ± 1.12	5.61 ± 1.10	+22.1%	0.071	1.05 (-0.15 – 2.18)
άλφα (8-13 Hz)	Ινιακή (O1-O2)	7.42 ± 1.02	8.73 ± 1.18	−15.0%	0.083	0.97 (-0.22 – 2.05)
βήτα (13-30 Hz)	Κεντρική (C3-C4)	3.14 ± 0.74	2.98 ± 0.68	+5.4%	0.312	0.23 (-0.91 – 1.30)
γάμμα (30-50 Hz)	Βρεγματική (P3-P4)	1.86 ± 0.50	1.91 ± 0.46	−2.6%	0.548	0.10 (-1.03 – 1.18)

Συγκεκριμένα, στην Ομάδα 2 παρατηρήθηκε αυξημένη δραστηριότητα στη ζώνη άλφα στους ινιακούς και βρεγματικούς λοβούς, όπως αναμενόταν λόγω της παρουσίας οπτικών ερεθισμάτων και της ενεργοποίησης των περιοχών που σχετίζονται με την οπτική επεξεργασία. Αντίθετα, στην ομάδα με παρεμπόδιση όρασης (Ομάδα 1) η ισχύς στη ζώνη άλφα ήταν μειωμένη, ενώ παρατηρήθηκε αύξηση της ισχύος στη ζώνη θήτα κυρίως στις μετωπιαίες περιοχές, υποδηλώνοντας εντονότερη γνωστική προσπάθεια και αυξημένη ενεργοποίηση μη οπτικών συστημάτων προσανατολισμού. Οι διαφορές αυτές επέδειξαν στατιστική σημαντικότητα, κάτι που (σε αντίθεση με τις ζώνες άλφα και θήτα) δεν παρατηρήθηκε για τις ζώνες δέλτα, βήτα, και γάμμα. Σε αυτές, παρατηρήθηκε ήπια αύξηση της ισχύος για την ομάδα με οπτική παρεμπόδιση στις ζώνες βήτα και δέλτα, ενώ στις ζώνη γάμμα πραγματοποιήθηκε μικρή μείωση. Παρόλα αυτά, τα αποτελέσματα αυτά δεν μπορούν να στοιχειοθετήσουν κάποια αξιόπιστη νευροφυσιολογική ερμηνεία, καθώς δεν παρουσιάζουν στατιστική σημαντικότητα.

Συνολικά, τα δεδομένα υποστηρίζουν την ύπαρξη λειτουργικών διαφοροποιήσεων στην κατανομή της φασματικής ισχύος ανάμεσα στις συνθήκες με και χωρίς οπτική είσοδο, ενώ οι οριακές στατιστικές τιμές αποδίδονται στο περιορισμένο πλήθος συμμετεχόντων.

Επιπλέον, σε συμπεριφορικό επίπεδο, καταγράφηκε για κάθε συμμετέχοντα ο χρόνος απόκρισης, προκειμένου να πραγματοποιηθεί σύγκριση μεταξύ των ομάδων. Όπως ήταν αναμενόμενο, τα αποτελέσματα έδειξαν μικρότερο χρόνο απόκρισης για την Ομάδα 1, κάτι που μπορεί να αποδοθεί στη μεγαλύτερη αυτοπεποίθηση κατά την απόκριση λόγω της οπτικής κωδικοποίησης του πειραματικού χώρου.

10. Συμπεράσματα

Η παρούσα εργασία διερεύνησε πώς διαφοροποιείται η εγκεφαλική δραστηριότητα κατά την κωδικοποίηση και πλοήγηση στον χώρο υπό συνθήκη παρεμπόδισης όρασης. Στο πλαίσιο αυτό, η σύγκριση των δύο ομάδων ανέδειξε εντονότερη δραστηριότητα ζώνης θ στις μετωπιαίες θέσεις στην ομάδα με παρεμπόδιση όρασης και εντονότερη δραστηριότητα ζώνης α σε ινιακές και βρεγματικές περιοχές για την ομάδα χωρίς παρεμπόδιση όρασης. Συμπεριφορικά, παρατηρήθηκαν επίσης διαφορές στον χρόνο απόκρισης μεταξύ των ομάδων κατά την πειραματική διαδικασία. Προφανώς, οι συμπεριφορικές διαφορές μπορούν να ερμηνευθούν υπό το πρίσμα ότι η οπτική κωδικοποίηση διευκολύνει τη μετέπειτα απτική πλοήγηση, άρα μειώνει τις απαιτήσεις γνωσιακής επεξεργασίας κατά τη φάση της απόκρισης.

Εάν επιχειρηθεί η ερμηνεία των στατιστικά σημαντικών αποτελεσμάτων της ανάλυσης του σήματος ηλεκτροεγκεφαλογραφήματος, η ενίσχυση στη δραστηριότητα της ζώνης θ υπό την απουσία όρασης είναι σύμφωνη με την καθιερωμένη συσχέτιση της συγκεκριμένης ζώνης με τη γνωστική προσπάθεια, και τη διατήρηση/ανάκληση χωρικής πληροφορίας, ειδικά όταν απαιτείται πολυαισθητηριακή εσωτερική χαρτογράφηση (σε σχέση με την απλή οπτική κωδικοποίηση) [[91], [92]]. Όσον αφορά τις διαφορές που παρατηρήθηκαν στη ζώνη α, η χαμηλότερη ισχύς ζώνης υπό τη συνθήκη παρεμπόδισης όρασης μπορεί να σχετίζεται με τη μειωμένη εμπλοκή του οπτικού φλοιού κατά την απτική χαρτογράφηση [[93], [94]]. Αξίζει επίσης να σημειωθεί ότι παρότι οι διαφορές αυτές είναι οριακά στατιστικά σημαντικές, οι αντίστοιχες τιμές Cohen’s d παρουσιάζουν μεγάλο εύρος διαστήματος εμπιστοσύνης, κάτι που προφανώς αποδίδεται στο πολύ μικρό δείγμα πληθυσμού που χρησιμοποιήθηκε κατά την πειραματική διαδικασία.

Σε γενικές γραμμές, τα αποτελέσματα αυτά δείχνουν ότι η πλοήγηση χωρίς όραση στρατολογεί εντονότερα μετωπιαία/εκτελεστικά δίκτυα και περιορίζει την επεξεργασία (σε σχέση με τη ζώνη α) σε ινιακές περιοχές, γεγονός που μπορεί να έχει πρακτική αξία για εκπαίδευση/αποκατάσταση ατόμων με οπτικές δυσκολίες βάσει βιοδεικτών ηλεκτροεγκεφαλογραφήματος και στο πλαίσιο σχεδιασμού και εφαρμογής διεπαφών εγκεφάλου υπολογιστή (BCI-s) για καθοδήγηση σε περιβάλλοντα μειωμένης όρασης.

Παρόλα αυτά, είναι σημαντικό να αναφερθούν συγκεκριμένοι περιορισμοί που σχετίζονται τόσο με τη διεξαγωγή του πειράματος όσο και με την ανάλυση και την ερμηνεία των δεδομένων αυτού. Καταρχάς, το μικρό μέγεθος δείγματος περιορίζει την στατιστική ισχύ των αποτελεσμάτων και την ικανότητα γενίκευσης, η οποία απαιτεί τη διεξαγωγή μελέτης με πολύ μεγαλύτερο πλήθος συμμετεχόντων. Επιπλέον, η εργασία επικεντρώθηκε στην ανάπτυξη του πειραματικού πρωτοκόλλου και της διαδικασίας για την εκτέλεση του πειράματος, χρησιμοποιώντας βασικές μετρικές για την ανάλυση των σημάτων ΗΕΓ. Η χρήση περισσότερων χαρακτηριστικών (πχ προκλητών δυναμικών) θα μπορούσε να δώσει την ευκαιρία για περισσότερα συμπεράσματα σχετικά με τις υποβόσκουσες διεργασίες. Σε τεχνικό επίπεδο, η χρήση της «ξηρής» καταγραφής μέσω του φορητού εγκεφαλογράφου είναι αρκετά ευαίσθητη στην επίδραση θορύβου και δεν επιτυγχάνει τη βέλτιστη επαφή μεταξύ των ηλεκτροδίων και της επιφάνειας της κεφαλής. Στο πλαίσιο αυτό, η χρήση «υγρής» καταγραφής θα μπορούσε να δώσει σήμα υψηλότερης ποιότητας (δηλ. υψηλότερο σηματοθορυβικό λόγο - SNR), ενσωματώνοντας και μεγαλύτερο πλήθος ηλεκτροδίων. Ένα τέτοιο πείραμα θα διευκόλυνε επίσης την εφαρμογή περισσότερων τεχνικών εξαγωγής χαρακτηριστικών, με δυνατότητα ανάλυσης και σε επίπεδο πηγών εγκεφαλικής δραστηριότητας πέρα από την ανάλυση σε επίπεδο επιφανειακών αισθητήρων.

Τέλος, πρέπει να σημειωθεί ότι το παρόν πείραμα σχεδιάστηκε για στατικές συνθήκες και προσανατολισμό σε δισδιάστατο πλέγμα με χρήση απλών κινήσεων των χεριών, επομένως τα

συμπεράσματα -αν και αντικατοπτρίζουν εγκεφαλική δραστηριότητα που σχετίζεται με τη χωρική κωδικοποίηση και τον προσανατολισμό- δεν μπορούν να γενικευθούν για πραγματικό πλαίσιο συνολικής κίνησης (πχ βάδισης) με εργασία προσανατολισμού σε τρισδιάστατο χώρο. Σε αυτή την περίπτωση, αναμένεται να παρατηρηθούν διαφορές. Υπάρχει σημαντικό πλήθος μελετώ που καταδεικνύουν ευρεία ανακατανομή της εγκεφαλικής δραστηριότητας υπό συνθήκες προσωρινής απουσίας όρασης. Για παράδειγμα, η απτική επεξεργασία ενεργοποιεί τον πρωτογενή και δευτερογενή οπτικό φλοιό που εδρεύπυν στον ινιακό λοβό, ενώ συνοδεύεται από αυξημένη συμμετοχή των βρεγματικών και μετωπιαίων περιοχών, αντανακλώντας τη μετάβαση σε πολυαισθητηριακή χαρτογράφιση [95]. Αντίστοιχα κατά την πλοήγηση σε τρισδιάστατο χώρο, η εγκεφαλική δραστηριότητα διαφοροποιείται ανάλογα με το αναφορικό πλαίσιο. Συγκεκριμένα, οι βρεγματοϊνιακές περιοχές υποστηρίζουν εγωκεντρική οργάνωση του χώρου ενώ ο ιππόκαμπος και οι παραϊπποκαμπικές δομές ενισχύουν την αλλοκεντρική αναπαράσταση [96]. Επομένως, η παρούσα εργασία αποτέλεσε ένα αρχικό βήμα προς την ανάπτυξη ενός ολοκληρωμένου πρωτοκόλλου μελέτης των διεργασιών χωρικής κωδικοποίησης και προσανατολισμού βάσει της εγκεφαλικής δραστηριότητας, το οποίο θα είναι εφαρμόσιμο σε πραγματικές συνθήκες τρισδιάστατου πλαισίου αξιοποιώντας το πρωτόκολλο που αναπτύχθηκε και δοκιμάστηκε πιλοτικά σε πλαίσιο δύο διαστάσεων.

11. Βιβλιογραφία

[1] ‘Κεντρικό νευρικό σύστημα’, *Βικιπαίδεια*. 7 Φεβρουάριος 2020. Ημερομηνία πρόσβασης: 13 Νοέμβριος 2021. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: https://el.wikipedia.org/w/index.php?title=%CE%9A%CE%B5%CE%BD%CF%84%CF%81%CE%B9%CE%BA%CF%8C_%CE%BD%CE%B5%CF%85%CF%81%CE%B9%CE%BA%CF%8C_%CF%83%CF%8D%CF%83%CF%84%CE%B7%CE%BC%CE%B1&oldid=8040001

[2] ‘Εγκεφαλικός φλοιός’, *Βικιπαίδεια*. 12 Σεπτέμβριος 2018. Ημερομηνία πρόσβασης: 13 Νοέμβριος 2021. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: https://el.wikipedia.org/w/index.php?title=%CE%95%CE%B3%CE%BA%CE%B5%CF%86%CE%B1%CE%BB%CE%B9%CE%BA%CF%8C%CF%82_%CF%86%CE%BB%CE%BF%CE%B9%CF%8C%CF%82&oldid=7213131

[3] ‘What Are the Different Parts of the Brain?’, Verywell Mind. Ημερομηνία πρόσβασης: 13 Νοέμβριος 2021. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: https://www.verywellmind.com/the-anatomy-of-the-brain-2794895

[4] ‘Functional Areas of The Cerebral Cortex’. Ημερομηνία πρόσβασης: 13 Νοέμβριος 2021. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: https://antranik.org/functional-areas-of-the-cerebral-cortex/

[5] ‘Functional Systems of the Cerebral Cortex | Boundless Anatomy and Physiology’. Ημερομηνία πρόσβασης: 13 Νοέμβριος 2021. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: https://courses.lumenlearning.com/boundless-ap/chapter/functional-systems-of-the-cerebral-cortex/

[6] K. Zilles και K. Amunts, ‘Centenary of Brodmann’s map — conception and fate’, *Nat Rev Neurosci*, τ. 11, τχ. 2, σσ. 139–145, Φεβρουαρίου 2010, doi: 10.1038/nrn2776.

[7] C. Hönegger, C. Atteneder, B. Griesmayr, E. Holz, E. Weber, και P. Sauseng, ‘Neural correlates of visuo-spatial working memory encoding—An EEG study’, *Neuroscience Letters*, τ. 500, τχ. 2, σσ. 118–122, Αυγούστου 2011, doi: 10.1016/j.neulet.2011.06.017.

[8] ‘Spatial memory’, *Wikipedia*. 28 Οκτώβριος 2021. Ημερομηνία πρόσβασης: 13 Νοέμβριος 2021. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: https://en.wikipedia.org/w/index.php?title=Spatial_memory&oldid=1052338058

[9] M. van Asselen, R. P. C. Kessels, S. F. W. Neggers, L. J. Kappelle, C. J. M. Frijns, και A. Postma, ‘Brain areas involved in spatial working memory’, *Neuropsychologia*, τ. 44, τχ. 7, σσ. 1185–1194, Ιανουαρίου 2006, doi: 10.1016/j.neuropsychologia.2005.10.005.

[10] ‘Spatial disorientation’, *Wikipedia*. 5 Οκτώβριος 2021. Ημερομηνία πρόσβασης: 17 Νοέμβριος 2021. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: https://en.wikipedia.org/w/index.php?title=Spatial_disorientation&oldid=1048328583

[11] A. Berthoz και I. Viaud-Delmon, ‘Multisensory integration in spatial orientation’, *Current Opinion in Neurobiology*, τ. 9, τχ. 6, σσ. 708–712, Δεκεμβρίου 1999, doi: 10.1016/S0959-4388(99)00041-0.

[12] ‘What Is Proprioception?’, WebMD. Ημερομηνία πρόσβασης: 16 Νοέμβριος 2021. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: https://www.webmd.com/brain/what-is-proprioception

[13] R. Maxwell, ‘Spatial Orientation and the Brain: The Effects of Map Reading and Navigation’, GIS Lounge. Ημερομηνία πρόσβασης: 13 Νοέμβριος 2021. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: https://www.gislounge.com/spatial-orientation-and-the-brain-the-effects-of-map-reading-and-navigation/

[14] M. J. Sholl, ‘Landmarks, places, environments: Multiple mind—Brain systems for spatial orientation’, *Geoforum*, τ. 23, τχ. 2, σσ. 151–164, Μαΐου 1992, doi: 10.1016/0016-7185(92)90013-T.

[15] ‘CogniFit’, Spatial Perception- Cognitive Ability. Ημερομηνία πρόσβασης: 13 Νοέμβριος 2021. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: https://www.cognifit.com/science/cognitive-skills/spatial-perception

[16] G. Galati, G. Pelle, A. Berthoz, και G. Committeri, ‘Multiple reference frames used by the human brain for spatial perception and memory’, *Exp Brain Res*, τ. 206, τχ. 2, σσ. 109–120, Οκτωβρίου 2010, doi: 10.1007/s00221-010-2168-8.

[17] G. Vallar και A. Maravita, ‘Personal and Extrapersonal Spatial Perception’, στο *Handbook of Neuroscience for the Behavioral Sciences*, G. G. Berntson και J. T. Cacioppo, Επιμ., Hoboken, NJ, USA: John Wiley & Sons, Inc., 2009, σ. neubb001016. doi: 10.1002/9780470478509.neubb001016.

[18] N. A. Herweg και M. J. Kahana, ‘Spatial Representations in the Human Brain’, *Front. Hum. Neurosci.*, τ. 12, σ. 297, Ιουλίου 2018, doi: 10.3389/fnhum.2018.00297.

[19] ‘Όραση’, *Βικιπαίδεια*. 24 Φεβρουάριος 2021. Ημερομηνία πρόσβασης: 14 Νοέμβριος 2021. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: https://el.wikipedia.org/w/index.php?title=%CE%8C%CF%81%CE%B1%CF%83%CE%B7&oldid=8702557
[20] S. Millar, ‘Models of Sensory Deprivation: The Nature/Nurture Dichotomy and Spatial Representation in the Blind’, *International Journal of Behavioral Development*, τ. 11, τχ. 1, σσ. 69–87, Μαρτίου 1988, doi: 10.1177/016502548801100105.
[21] P. Voss, ‘Auditory Spatial Perception without Vision’, *Front. Psychol.*, τ. 07, Δεκεμβρίου 2016, doi: 10.3389/fpsyg.2016.01960.
[22] M. Jourdain, *Diderot’s Early Philosophical Works*. 1916. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: https://books.google.gr/books?id=0rnPe1ofuEYC
[23] W. James, *Principles of psychology. Volumes I and II Volumes I and II*. 2018.
[24] L. Rinaldi, L. B. Merabet, T. Vecchi, και Z. Cattaneo, ‘The spatial representation of number, time, and serial order following sensory deprivation: A systematic review’, *Neuroscience & Biobehavioral Reviews*, τ. 90, σσ. 371–380, Ιουλίου 2018, doi: 10.1016/j.neubiorev.2018.04.021.
[25] ‘Αφή’, *Βικιπαίδεια*. 7 Μάρτιος 2021. Ημερομηνία πρόσβασης: 18 Νοέμβριος 2021. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: https://el.wikipedia.org/w/index.php?title=%CE%91%CF%86%CE%AE&oldid=8721126
[26] S. J. Lederman, R. L. Klatzky, και P. O. Barber, ‘Spatial and movement-based heuristics for encoding pattern information through touch.’, *Journal of Experimental Psychology: General*, τ. 114, τχ. 1, σσ. 33–49, 1985, doi: 10.1037/0096-3445.114.1.33.
[27] K. Papadopoulos και E. Koustriava, ‘The impact of vision in spatial coding’, *Research in Developmental Disabilities*, τ. 32, τχ. 6, σσ. 2084–2091, Νοεμβρίου 2011, doi: 10.1016/j.ridd.2011.07.041.
[28] K. Papadopoulos, E. Koustriava, και L. Kartasidou, ‘Spatial Coding of Individuals With Visual Impairments’, *J Spec Educ*, τ. 46, τχ. 3, σσ. 180–190, Νοεμβρίου 2012, doi: 10.1177/0022466910383016.
[29] R. Carter *κ.ά.*, *The human brain book*, 1st American ed. London ; New York, N.Y: DK Pub, 2009.
[30] M.-M. Mesulam, Επιμ., *Principles of behavioral and cognitive neurology*, 2nd ed. Oxford ; New York: Oxford University Press, 2000.
[31] S. Hrishikesan, ‘Advantages, Disadvantages and Applications of EEG’, Electronics and Communication Study Materials. Ημερομηνία πρόσβασης: 15 Νοέμβριος 2021. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: https://www.electronicsandcommunications.com/2018/08/advantages-disadvantages-applications-of-eeg.html
[32] SpinalCord.com, ‘Limbic Lobe | SpinalCord.com’. Ημερομηνία πρόσβασης: 15 Νοέμβριος 2021. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: https://www.spinalcord.com/limbic-lobe
[33] K. Brodmann και L. J. Gary, *Brodmann’s localisation in the cerebral cortex: the principles of comparative localisation in the cerebral cortex based on cytoarchitectonics*. New York, NY: Springer, 2006.
[34] ‘Brodmann’s Interactive Atlas’. Ημερομηνία πρόσβασης: 15 Νοέμβριος 2021. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: http://www.fmriconsulting.com/brodmann/Introduction.html
[35] W. Penfield και E. Boldrey, ‘SOMATIC MOTOR AND SENSORY REPRESENTATION IN THE CEREBRAL CORTEX OF MAN AS STUDIED BY ELECTRICAL STIMULATION’, *Brain*, τ. 60, τχ. 4, σσ. 389–443, 1937, doi: 10.1093/brain/60.4.389.
[36] *Exercise-Cognition Interaction.* Elsevier Science, 2015. Ημερομηνία πρόσβασης: 22 Νοέμβριος 2021. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: http://www.totalboox.com/book/id-1759072832684219413
[37] J. Polich, ‘P300 as a clinical assay: rationale, evaluation, and findings’, *International Journal of Psychophysiology*, τ. 38, τχ. 1, σσ. 3–19, Οκτωβρίου 2000, doi: 10.1016/S0167-8760(00)00127-6.
[38] ‘Alpha wave’, *Wikipedia*. 9 Νοέμβριος 2021. Ημερομηνία πρόσβασης: 15 Νοέμβριος 2021. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: https://en.wikipedia.org/w/index.php?title=Alpha_wave&oldid=1054270503
[39] R. G. Grenell, ‘*The Cerebral Cortex of Man. A Clinical Study of Localization of Function.* Wilder Penfield , Theodore Rasmussen *Essay on the Cerebral Cortex. A Monograph in American Lectures in Anatomy.* Gerhardt von Bonin’, *The Quarterly Review of Biology*, τ. 27, τχ. 1, σσ. 108–109, Μαρτίου 1952, doi: 10.1086/398778.

[40] ‘Beta wave’, *Wikipedia*. 9 Νοέμβριος 2021. Ημερομηνία πρόσβασης: 15 Νοέμβριος 2021. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: https://en.wikipedia.org/w/index.php?title=Beta_wave&oldid=1054271549
[41] ‘THE BRAIN FROM TOP TO BOTTOM’. Ημερομηνία πρόσβασης: 15 Νοέμβριος 2021. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: https://thebrain.mcgill.ca/flash/d/d_07/d_07_p/d_07_p_tra/d_07_p_tra.html
[42] ‘Delta wave’, *Wikipedia*. 28 Ιούλιος 2021. Ημερομηνία πρόσβασης: 15 Νοέμβριος 2021. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: https://en.wikipedia.org/w/index.php?title=Delta_wave&oldid=1035959038
[43] ‘The Working Memory Model by Baddeley and Hitch’. Ημερομηνία πρόσβασης: 15 Νοέμβριος 2021. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: https://explorable.com/working-memory-model
[44] T. Harmony *κ.ά.*, ‘EEG delta activity: an indicator of attention to internal processing during performance of mental tasks’, *International Journal of Psychophysiology*, τ. 24, τχ. 1, σσ. 161–171, Νοεμβρίου 1996, doi: 10.1016/S0167-8760(96)00053-0.
[45] S. Dimitriadis, N. Laskaris, V. Tsirka, M. Vourkas, και M. Sifis, ‘What does delta band tell us about cognitive processes: A mental calculation study’, *Neuroscience letters*, τ. 483, σσ. 11–5, Οκτωβρίου 2010, doi: 10.1016/j.neulet.2010.07.034.
[46] ‘Working Memory Model | Simply Psychology’. Ημερομηνία πρόσβασης: 15 Νοέμβριος 2021. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: https://www.simplypsychology.org/working%20memory.html
[47] ‘Theta wave’, *Wikipedia*. 6 Ιούλιος 2021. Ημερομηνία πρόσβασης: 15 Νοέμβριος 2021. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: https://en.wikipedia.org/w/index.php?title=Theta_wave&oldid=1032191208
[48] ‘The Human Memory | What It Is, How It Works & How It Can Go Wrong’. Ημερομηνία πρόσβασης: 15 Νοέμβριος 2021. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: https://human-memory.net/
[49] M. Eimer, A. Gosling, S. Nicholas, και M. Kiss, ‘The N170 component and its links to configural face processing: A rapid neural adaptation study’, *Brain Research*, τ. 1376, σσ. 76–87, Φεβρουαρίου 2011, doi: 10.1016/j.brainres.2010.12.046.
[50] A. Gosling και M. Eimer, ‘An event-related brain potential study of explicit face recognition’, *Neuropsychologia*, τ. 49, τχ. 9, σσ. 2736–2745, Ιουλίου 2011, doi: 10.1016/j.neuropsychologia.2011.05.025.
[51] M. Eimer, ‘The N2pc component as an indicator of attentional selectivity’, *Electroencephalography and Clinical Neurophysiology*, τ. 99, τχ. 3, σσ. 225–234, Σεπτεμβρίου 1996, doi: 10.1016/0013-4694(96)95711-9.
[52] M. Eimer, ‘Does the face-specific N170 component reflect the activity of a specialized eye processor?’:, *NeuroReport*, τ. 9, τχ. 13, σσ. 2945–2948, Σεπτεμβρίου 1998, doi: 10.1097/00001756-199809140-00005.
[53] G. Thierry, C. D. Martin, P. Downing, και A. J. Pegna, ‘Controlling for interstimulus perceptual variance abolishes N170 face selectivity’, *Nat Neurosci*, τ. 10, τχ. 4, σσ. 505–511, Απριλίου 2007, doi: 10.1038/nn1864.
[54] T. Marzi και M. P. Viggiano, ‘Deep and shallow encoding effects on face recognition: An ERP study’, *International Journal of Psychophysiology*, τ. 78, τχ. 3, σσ. 239–250, Δεκεμβρίου 2010, doi: 10.1016/j.ijpsycho.2010.08.005.
[55] K. E. Crowley και I. M. Colrain, ‘A review of the evidence for P2 being an independent component process: age, sleep and modality’, *Clinical Neurophysiology*, τ. 115, τχ. 4, σσ. 732–744, Απριλίου 2004, doi: 10.1016/j.clinph.2003.11.021.
[56] ‘Research Use of Emotiv EPOC’. Ημερομηνία πρόσβασης: 22 Νοέμβριος 2021. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: http://neurofeedback.visaduma.info/emotivresearch.htm
[57] M. Akin, ‘Comparison of Wavelet Transform and FFT Methods in the Analysis of EEG Signals’, *Journal of Medical Systems*, τ. 26, τχ. 3, σσ. 241–247, 2002, doi: 10.1023/A:1015075101937.
[58] E. Donchin και M. G. H. Coles, ‘Is the P300 component a manifestation of context updating?’, *Behav Brain Sci*, τ. 11, τχ. 03, σ. 357, Σεπτεμβρίου 1988, doi: 10.1017/S0140525X00058027.
[59] D. Ravden και J. Polich, ‘On P300 measurement stability: habituation, intra-trial block variation, and ultradian rhythms’, *Biological Psychology*, τ. 51, τχ. 1, σσ. 59–76, Οκτωβρίου 1999, doi: 10.1016/S0301-0511(99)00015-0.

[60] J. Polich, 'On the relationship between EEG and P300: individual differences, aging, and ultradian rhythms', *International Journal of Psychophysiology*, τ. 26, τχ. 1–3, σσ. 299–317, Ιουνίου 1997, doi: 10.1016/S0167-8760(97)00772-1.
[61] 'Independent component analysis', *Wikipedia*. 9 Νοέμβριος 2021. Ημερομηνία πρόσβασης: 24 Νοέμβριος 2021. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: https://en.wikipedia.org/w/index.php?title=Independent_component_analysis&oldid=1054394585
[62] U. Friese, M. Köster, U. Hassler, U. Martens, N. Trujillo-Barreto, και T. Gruber, 'Successful memory encoding is associated with increased cross-frequency coupling between frontal theta and posterior gamma oscillations in human scalp-recorded EEG', *NeuroImage*, τ. 66, σσ. 642–647, Φεβρουαρίου 2013, doi: 10.1016/j.neuroimage.2012.11.002.
[63] B. M. Roberts, L.-T. Hsieh, και C. Ranganath, 'Oscillatory activity during maintenance of spatial and temporal information in working memory', *Neuropsychologia*, τ. 51, τχ. 2, σσ. 349–357, Ιανουαρίου 2013, doi: 10.1016/j.neuropsychologia.2012.10.009.
[64] S. Xie, D. Kaiser, και R. M. Cichy, 'Visual Imagery and Perception Share Neural Representations in the Alpha Frequency Band', *Current Biology*, τ. 30, τχ. 13, σσ. 2621-2627.e5, Ιουλίου 2020, doi: 10.1016/j.cub.2020.04.074.
[65] J. E. Lisman και M. A. P. Idiart, 'Storage of 7 ± 2 Short-Term Memories in Oscillatory Subcycles', *Science*, τ. 267, τχ. 5203, σσ. 1512–1515, Μαρτίου 1995, doi: 10.1126/science.7878473.
[66] O. Jensen και C. D. Tesche, 'Frontal theta activity in humans increases with memory load in a working memory task', *Eur J of Neuroscience*, τ. 15, τχ. 8, σσ. 1395–1399, Απριλίου 2002, doi: 10.1046/j.1460-9568.2002.01975.x.
[67] A. Gevins, 'High-resolution EEG mapping of cortical activation related to working memory: effects of task difficulty, type of processing, and practice', *Cerebral Cortex*, τ. 7, τχ. 4, σσ. 374–385, Ιουνίου 1997, doi: 10.1093/cercor/7.4.374.
[68] M. Plank, H. J. Müller, J. Onton, S. Makeig, και K. Gramann, 'Human EEG Correlates of Spatial Navigation within Egocentric and Allocentric Reference Frames', στο *Spatial Cognition VII*, τ. 6222, C. Hölscher, T. F. Shipley, M. Olivetti Belardinelli, J. A. Bateman, και N. S. Newcombe, Επιμ., στο Lecture Notes in Computer Science, vol. 6222. , Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, σσ. 191–206. doi: 10.1007/978-3-642-14749-4_18.
[69] P. Sauseng *κ.ά.*, 'A shift of visual spatial attention is selectively associated with human EEG alpha activity', *Eur J of Neuroscience*, τ. 22, τχ. 11, σσ. 2917–2926, Δεκεμβρίου 2005, doi: 10.1111/j.1460-9568.2005.04482.x.
[70] G. Santaniello, M. Sebastián, L. Carretié, U. Fernández-Folgueiras, και J. A. Hinojosa, 'Haptic recognition memory following short-term visual deprivation: Behavioral and neural correlates from ERPs and alpha band oscillations', *Biological Psychology*, τ. 133, σσ. 18–29, Μαρτίου 2018, doi: 10.1016/j.biopsycho.2018.01.008.
[71] W. Klimesch, P. Sauseng, και S. Hanslmayr, 'EEG alpha oscillations: The inhibition–timing hypothesis', *Brain Research Reviews*, τ. 53, τχ. 1, σσ. 63–88, Ιανουαρίου 2007, doi: 10.1016/j.brainresrev.2006.06.003.
[72] D. J. Hawellek, I. M. Schepers, B. Roeder, A. K. Engel, M. Siegel, και J. F. Hipp, 'Altered Intrinsic Neuronal Interactions in the Visual Cortex of the Blind', *J. Neurosci.*, τ. 33, τχ. 43, σσ. 17072–17080, Οκτωβρίου 2013, doi: 10.1523/JNEUROSCI.1625-13.2013.
[73] A. Kriegseis, E. Hennighausen, F. Rösler, και B. Röder, 'Reduced EEG alpha activity over parieto-occipital brain areas in congenitally blind adults', *Clinical Neurophysiology*, τ. 117, τχ. 7, σσ. 1560–1573, Ιουλίου 2006, doi: 10.1016/j.clinph.2006.03.030.
[74] G. Pfurtscheller και F. H. Lopes Da Silva, 'Event-related EEG/MEG synchronization and desynchronization: basic principles', *Clinical Neurophysiology*, τ. 110, τχ. 11, σσ. 1842–1857, Νοεμβρίου 1999, doi: 10.1016/S1388-2457(99)00141-8.
[75] S. E. Levick, B. E. Wexler, T. Lorig, R. E. Gur, R. C. Gur, και G. E. Schwartz, 'Asymmetrical Visual Deprivation: A Technique to Differentially Influence Lateral Hemispheric Function', *Percept Mot Skills*, τ. 76, τχ. 3_suppl, σσ. 1363–1382, Ιουνίου 1993, doi: 10.2466/pms.1993.76.3c.1363.
[76] M. G. Saunders και J. P. Zubek, 'EEG changes in perceptual and sensory deprivation', *Electroencephalogr Clin Neurophysiol*, σ. Suppl 25:246+, 1967.

[77] A. Aldridge et al, 'Accessible Electroencephalograms (EEGs): A Comparative Review with OpenBCI's Ultracortex Mark IV Headset', παρουσιάστηκε στο 2019 29th International Conference Radioelektronika (RADIOELEKTRONIKA), Απριλίου 2019. doi: 10.1109/RADIOELEK.2019.8733482.
[78] M. Firdaus, N. Sathees Kumar, και A. Syed Faiz, 'An Approach In Determining Fatigueness And Drowsiness Detection Using EEG', Οκτωβρίου 2018, doi: 10.13140/RG.2.2.12630.50243.
[79] P. M. R. Reis, F. Hebenstreit, F. Gabsteiger, V. von Tscharner, και M. Lochmann, 'Methodological aspects of EEG and body dynamics measurements during motion', *Frontiers in Human Neuroscience*, τ. 8, τχ. 2014, Μαρτίου 2014, doi: https://doi.org/10.3389/fnhum.2014.00156.
[80] K. E. Mathewson, T. J. L. Harrison, και S. A. D. Kizuk, 'High and dry? Comparing active dry EEG electrodes to active and passive wet electrodes', *Psychophysiology*, τ. 54, τχ. 1, σσ. 74–82, Ιανουαρίου 2017, doi: 10.1111/psyp.12536.
[81] J. W. Y. Kam *κ.ά.*, 'Systematic comparison between a wireless EEG system with dry electrodes and a wired EEG system with wet electrodes', *NeuroImage*, τ. 184, σσ. 119–129, Ιανουαρίου 2019, doi: 10.1016/j.neuroimage.2018.09.012.
[82] P. Cattarello και R. Merletti, 'Characterization of dry and wet Electrode-Skin interfaces on different skin treatments for HDsEMG', στο *2016 IEEE International Symposium on Medical Measurements and Applications (MeMeA)*, Benevento, Italy: IEEE, Μαΐου 2016, σσ. 1–6. doi: 10.1109/MeMeA.2016.7533808.
[83] Y. Higashi, Y. Yokota, και Y. Naruse, 'Signal correlation between wet and original dry electrodes in electroencephalogram according to the contact impedance of dry electrodes', στο *2017 39th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, Jeju, Korea (South): IEEE, Ιουλίου 2017, σσ. 1062–1065. doi: 10.1109/EMBC.2017.8037010.
[84] S. L. Kappel και P. Kidmose, 'Study of impedance spectra for dry and wet EarEEG electrodes', στο *2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, Milan: IEEE, Αυγούστου 2015, σσ. 3161–3164. doi: 10.1109/EMBC.2015.7319063.
[85] T. O. Zander *κ.ά.*, 'A Dry EEG-System for Scientific Research and Brain–Computer Interfaces', *Front. Neurosci.*, τ. 5, 2011, doi: 10.3389/fnins.2011.00053.
[86] B. Meneses-Claudio και A. Roman-Gonzalez, 'Study of the Brain Waves for the identification of the Basic Needs of Patients with Amyotrophic Lateral Sclerosis', στο *2018 Congreso Argentino de Ciencias de la Informática y Desarrollos de Investigación (CACIDI)*, Buenos Aires: IEEE, Νοεμβρίου 2018, σσ. 1–6. doi: 10.1109/CACIDI.2018.8584193.
[87] J. M. Qiu, M. A. Casey, και S. G. Diamond, 'Assessing Feedback Response With a Wearable Electroencephalography System', *Front. Hum. Neurosci.*, τ. 13, σ. 258, Ιουλίου 2019, doi: 10.3389/fnhum.2019.00258.
[88] V. Peterson, C. Galván, H. Hernández, και R. Spies, 'A feasibility study of a complete low-cost consumer-grade brain-computer interface system', *Heliyon*, τ. 6, τχ. 3, σ. e03425, Μαρτίου 2020, doi: 10.1016/j.heliyon.2020.e03425.
[89] S. Parab και S. Bhalerao, 'Choosing statistical test', *Int J Ayurveda Res*, τ. 1, τχ. 3, σ. 187, 2010, doi: 10.4103/0974-7788.72494.
[90] Student, 'The Probable Error of a Mean', *Biometrika*, τ. 6, τχ. 1, σ. 1, Μαρτίου 1908, doi: 10.2307/2331554.
[91] J. B. Caplan, J. R. Madsen, A. Schulze-Bonhage, R. Aschenbrenner-Scheibe, E. L. Newman, και M. J. Kahana, 'Human θ Oscillations Related to Sensorimotor Integration and Spatial Learning', *J. Neurosci.*, τ. 23, τχ. 11, σσ. 4726–4736, Ιουνίου 2003, doi: 10.1523/JNEUROSCI.23-11-04726.2003.
[92] D. J. Mitchell, N. McNaughton, D. Flanagan, και I. J. Kirk, 'Frontal-midline theta from the perspective of hippocampal "theta"', *Progress in Neurobiology*, τ. 86, τχ. 3, σσ. 156–185, Νοεμβρίου 2008, doi: 10.1016/j.pneurobio.2008.09.005.
[93] T. Ergenoglu, T. Demiralp, Z. Bayraktaroglu, M. Ergen, H. Beydagi, και Y. Uresin, 'Alpha rhythm of the EEG modulates visual detection performance in humans', *Cognitive Brain Research*, τ. 20, τχ. 3, σσ. 376–383, Αυγούστου 2004, doi: 10.1016/j.cogbrainres.2004.03.009.
[94] M. Steriade, P. Gloor, R. R. Llinás, F. H. Lopes Da Silva, και M.-M. Mesulam, 'Basic mechanisms of cerebral rhythmic activities', *Electroencephalography and Clinical Neurophysiology*, τ. 76, τχ. 6, σσ. 481–508, Δεκεμβρίου 1990, doi: 10.1016/0013-4694(90)90001-Z.

[95] L. B. Merabet *κ.ά.*, ‘Combined Activation and Deactivation of Visual Cortex During Tactile Sensory Processing’, *Journal of Neurophysiology*, τ. 97, τχ. 2, σσ. 1633–1641, Φεβρουαρίου 2007, doi: 10.1152/jn.00806.2006.
[96] M. Plank, H. J. Müller, J. Onton, S. Makeig, και K. Gramann, ‘Human EEG Correlates of Spatial Navigation within Egocentric and Allocentric Reference Frames’, στο *Spatial Cognition VII*, τ. 6222, C. Hölscher, T. F. Shipley, M. Olivetti Belardinelli, J. A. Bateman, και N. S. Newcombe, Επιμ., στο Lecture Notes in Computer Science, vol. 6222. , Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, σσ. 191–206. doi: 10.1007/978-3-642-14749-4_18.

12. Παράρτημα Α: Κώδικας PsychoPy

Εισαγωγή βιβλιοθηκών & Αρχικοποίηση

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
from __future__ import absolute_import, division

import psychopy
psychopy.useVersion('2021.2.3')


from psychopy import locale_setup
from psychopy import prefs
prefs.hardware['audioLib'] = 'ptb'
prefs.hardware['audioLatencyMode'] = '4'
from psychopy import sound, gui, visual, core, data, event, logging, clock, colors
from psychopy.constants import (NOT_STARTED, STARTED, PLAYING, PAUSED,
                                STOPPED, FINISHED, PRESSED, RELEASED, FOREVER)

import numpy as np  # whole numpy lib is available, prepend 'np.'
from numpy import (sin, cos, tan, log, log10, pi, average,
                   sqrt, std, deg2rad, rad2deg, linspace, asarray)
from numpy.random import random, randint, normal, shuffle, choice as randchoice
import os  # handy system and path functions
import sys  # to get file system encoding

from psychopy.hardware import keyboard

import serial
port = serial.Serial("COM6", baudrate=115200)
port.write(1)

# Ensure that relative paths start from the same directory as this script
_thisDir = os.path.dirname(os.path.abspath(__file__))
os.chdir(_thisDir)
```

Πληροφορίες Πειράματος

```
# Store info about the experiment session
psychopyVersion = '2021.2.3'
expName = 'spatialExp'  # from the Builder filename that created this script
expInfo = {'participant': '', 'session': '001'}
dlg = gui.DlgFromDict(dictionary=expInfo, sortKeys=False, title=expName)
if dlg.OK == False:
    core.quit()  # user pressed cancel
expInfo['date'] = data.getDateStr()  # add a simple timestamp
expInfo['expName'] = expName
expInfo['psychopyVersion'] = psychopyVersion

# Data file name stem = absolute path + name; later add .psyexp, .csv, .log, etc
filename = _thisDir + os.sep + u'data/%s_%s_%s' % (expInfo['participant'], expName,
expInfo['date'])
```

```
# An ExperimentHandler isn't essential but helps with data saving
thisExp = data.ExperimentHandler(name=expName, version='',
    extraInfo=expInfo, runtimeInfo=None,

originPath="D:\\Nextcloud\\3.Academics\\5.Διπλωματικές\\Τρέχουσες\\02.Vision_deprivation_Spa
tial_Orientation_Δανάη_Λιάκου\\Danae's_Files\\Thesis_Files\\Implementation\\experiment_new\\
spatialExp.py",
    savePickle=True, saveWideText=True,
    dataFileName=filename)
# save a log file for detail verbose info
logFile = logging.LogFile(filename+'.log', level=logging.EXP)
logging.console.setLevel(logging.WARNING)  # this outputs to the screen, not a file

endExpNow = False  # flag for 'escape' or other condition => quit the exp
frameTolerance = 0.001  # how close to onset before 'same' frame
```

Δομή Πειράματος

```
# Start Code - component code to be run after the window creation

# Setup the Window
win = visual.Window(
    size=[800, 480], fullscr=True, screen=1,
    winType='pyglet', allowGUI=True, allowStencil=False,
    monitor='testMonitor', color=[-1,-1,-1], colorSpace='rgb',
    blendMode='avg', useFBO=True,
    units='height')
# store frame rate of monitor if we can measure it
expInfo['frameRate'] = win.getActualFrameRate()
if expInfo['frameRate'] != None:
    frameDur = 1.0 / round(expInfo['frameRate'])
else:
    frameDur = 1.0 / 60.0  # could not measure, so guess

# Setup eyetracking
ioDevice = ioConfig = ioSession = ioServer = eyetracker = None

# create a default keyboard (e.g. to check for escape)
defaultKeyboard = keyboard.Keyboard()

# Initialize components for Routine "rest_init"
rest_initClock = core.Clock()
instructions = visual.TextStim(win=win, name='instructions',
    text='Μόλις ακούσετε ήχο, να ακολουθήσετε τις οδηγίες ανάλογα με την ομάδα στην οποία
ανήκετε:\n\n-Ομάδα 1: Θα εμφανιστεί ένας σταυρός στην οθόνη. Να κοιτάτε τον σταυρό
παραμένοντας σε ηρεμία μέχρι να ακούσετε ξανά ήχο.\n\n-Ομάδα 2: Κλείστε τα μάτια σας και
παραμείνετε σε ηρεμία μέχρι να ακούσετε ξανά ήχο.\n\nΠατήστε space όταν είστε έτοιμοι.',
    font='Open Sans',
    pos=(0, 0), height=0.05, wrapWidth=None, ori=0.0,
    color='white', colorSpace='rgb', opacity=None,
    languageStyle='LTR',
    depth=0.0);
key_resp = keyboard.Keyboard()

# Initialize components for Routine "rest"
```

```
restClock = core.Clock()
rest_go = sound.Sound('Censor Beep Sound Effect.wav', secs=1.0, stereo=True, hamming=True,
    name='rest_go')
rest_go.setVolume(1.0)
cross_rest = visual.ImageStim(
    win=win,
    name='cross_rest',
    image='images/create+cross+new+plus+icon-1320073176732969305_512.png', mask=None,
    ori=0.0, pos=(0, 0), size=(0.2, 0.2),
    color=[1,1,1], colorSpace='rgb', opacity=None,
    flipHoriz=False, flipVert=False,
    texRes=128.0, interpolate=True, depth=-1.0)
rest_stop = sound.Sound('Censor Beep Sound Effect.wav', secs=1.0, stereo=True, hamming=True,
    name='rest_stop')
rest_stop.setVolume(1.0)
ISI = clock.StaticPeriod(win=win, screenHz=expInfo['frameRate'], name='ISI')

# Initialize components for Routine "movement_init"
movement_initClock = core.Clock()
instructions_2 = visual.TextStim(win=win, name='instructions_2',
    text='Κρατήστε κλειστά τα μάτια σας. Μόλις ακούσετε ήχο, ξεκινήστε να κινείτε με αργές
κινήσεις το κυρίαρχο χέρι τυχαία πάνω στο ταμπλό.\n\nΠατήστε space όταν είστε έτοιμοι.',
    font='Open Sans',
    pos=(0, 0), height=0.05, wrapWidth=None, ori=0.0,
    color='white', colorSpace='rgb', opacity=None,
    languageStyle='LTR',
    depth=0.0);
key_resp_2 = keyboard.Keyboard()

# Initialize components for Routine "movement"
movementClock = core.Clock()
move_go = sound.Sound('Censor Beep Sound Effect.wav', secs=1.0, stereo=True, hamming=True,
    name='move_go')
move_go.setVolume(1.0)
move_stop = sound.Sound('Censor Beep Sound Effect.wav', secs=1.0, stereo=True, hamming=True,
    name='move_stop')
move_stop.setVolume(1.0)
ISI_2 = clock.StaticPeriod(win=win, screenHz=expInfo['frameRate'], name='ISI_2')

# Initialize components for Routine "encoding_init"
encoding_initClock = core.Clock()
instructions_3 = visual.TextStim(win=win, name='instructions_3',
    text='Μόλις ακούσετε ήχο, να ακολουθήσετε τις οδηγίες ανάλογα με την ομάδα στην οποία
ανήκετε:\n\n-Ομάδα 1: Παρατηρήστε το ταμπλό (χωρίς κίνηση κεφαλιού) και προσπαθήστε να
προσανατολιστείτε στις διάφορες θέσεις χωρίς να κινήσετε τα χέρια σας.\n\n-Ομάδα 2: Με
κλειστά μάτια και χωρίς κίνηση κεφαλιού, κινείστε το κυρίαρχο χέρι σας πάνω στο ταμπλό από
αριστερά προς τα δεξιά και ανά γραμμή. Προσπαθείστε να προσανατολιστείτε στις διάφορες
θέσεις.\n\nΠατήστε space όταν είστε έτοιμοι.',
    font='Open Sans',
    pos=(0, 0), height=0.05, wrapWidth=None, ori=0.0,
    color='white', colorSpace='rgb', opacity=None,
    languageStyle='LTR',
    depth=0.0);
key_resp_3 = keyboard.Keyboard()

# Initialize components for Routine "encoding"
encodingClock = core.Clock()
encoding_go = sound.Sound('Censor Beep Sound Effect.wav', secs=1.0, stereo=True,
hamming=True,
```

```
    name='encoding_go')
encoding_go.setVolume(1.0)
encoding_stop = sound.Sound('Censor Beep Sound Effect.wav', secs=1, stereo=True,
hamming=True,
    name='encoding_stop')
encoding_stop.setVolume(1.0)
button1_1 = visual.ButtonStim(win,
    text=None, font='Arvo',
    pos=[-300,180],units='pix',
    letterHeight=0.05,
    size=[120,120], borderWidth=1.0,
    fillColor='black', borderColor='white',
    color='white', colorSpace='rgb',
    opacity=None,
    bold=True, italic=False,
    padding=None,
    anchor='center',
    name='button1_1'
)
button1_1.buttonClock = core.Clock()
button1_2 = visual.ButtonStim(win,
    text=None, font='Arvo',
    pos=[-180,180],units='pix',
    letterHeight=0.05,
    size=[120,120], borderWidth=1.0,
    fillColor='black', borderColor='white',
    color='white', colorSpace='rgb',
    opacity=None,
    bold=True, italic=False,
    padding=None,
    anchor='center',
    name='button1_2'
)
button1_2.buttonClock = core.Clock()
button1_3 = visual.ButtonStim(win,
    text=None, font='Arvo',
    pos=[-60,180],units='pix',
    letterHeight=0.05,
    size=[120,120], borderWidth=1.0,
    fillColor='black', borderColor='white',
    color='white', colorSpace='rgb',
    opacity=None,
    bold=True, italic=False,
    padding=None,
    anchor='center',
    name='button1_3'
)
button1_3.buttonClock = core.Clock()
button1_4 = visual.ButtonStim(win,
    text=None, font='Arvo',
    pos=[60,180],units='pix',
    letterHeight=0.05,
    size=[120,120], borderWidth=1.0,
    fillColor='black', borderColor='white',
    color='white', colorSpace='rgb',
    opacity=None,
    bold=True, italic=False,
    padding=None,
    anchor='center',
```

```
    name='button1_4'
)
button1_4.buttonClock = core.Clock()
button1_5 = visual.ButtonStim(win,
    text=None, font='Arvo',
    pos=[180,180],units='pix',
    letterHeight=0.05,
    size=[120,120], borderWidth=1.0,
    fillColor='black', borderColor='white',
    color='white', colorSpace='rgb',
    opacity=None,
    bold=True, italic=False,
    padding=None,
    anchor='center',
    name='button1_5'
)
button1_5.buttonClock = core.Clock()
button1_6 = visual.ButtonStim(win,
    text=None, font='Arvo',
    pos=[300,180],units='pix',
    letterHeight=0.05,
    size=[120,120], borderWidth=1.0,
    fillColor='black', borderColor='white',
    color='white', colorSpace='rgb',
    opacity=None,
    bold=True, italic=False,
    padding=None,
    anchor='center',
    name='button1_6'
)
button1_6.buttonClock = core.Clock()
button2_1 = visual.ButtonStim(win,
    text=None, font='Arvo',
    pos=[-300,60],units='pix',
    letterHeight=0.05,
    size=[120,120], borderWidth=1.0,
    fillColor='black', borderColor='white',
    color='white', colorSpace='rgb',
    opacity=None,
    bold=True, italic=False,
    padding=None,
    anchor='center',
    name='button2_1'
)
button2_1.buttonClock = core.Clock()
button2_2 = visual.ButtonStim(win,
    text=None, font='Arvo',
    pos=[-180,60],units='pix',
    letterHeight=0.05,
    size=[120,120], borderWidth=1.0,
    fillColor='black', borderColor='white',
    color='white', colorSpace='rgb',
    opacity=None,
    bold=True, italic=False,
    padding=None,
    anchor='center',
    name='button2_2'
)
button2_2.buttonClock = core.Clock()
```

```
button2_3 = visual.ButtonStim(win,
   text=None, font='Arvo',
   pos=[-60,60],units='pix',
   letterHeight=0.05,
   size=[120,120], borderWidth=1.0,
   fillColor='black', borderColor='white',
   color='white', colorSpace='rgb',
   opacity=None,
   bold=True, italic=False,
   padding=None,
   anchor='center',
   name='button2_3'
)
button2_3.buttonClock = core.Clock()
button2_4 = visual.ButtonStim(win,
   text=None, font='Arvo',
   pos=[60,60],units='pix',
   letterHeight=0.05,
   size=[120,120], borderWidth=1.0,
   fillColor='black', borderColor='white',
   color='white', colorSpace='rgb',
   opacity=None,
   bold=True, italic=False,
   padding=None,
   anchor='center',
   name='button2_4'
)
button2_4.buttonClock = core.Clock()
button2_5 = visual.ButtonStim(win,
   text=None, font='Arvo',
   pos=[180,60],units='pix',
   letterHeight=0.05,
   size=[120,120], borderWidth=1.0,
   fillColor='black', borderColor='white',
   color='white', colorSpace='rgb',
   opacity=None,
   bold=True, italic=False,
   padding=None,
   anchor='center',
   name='button2_5'
)
button2_5.buttonClock = core.Clock()
button2_6 = visual.ButtonStim(win,
   text=None, font='Arvo',
   pos=[300,60],units='pix',
   letterHeight=0.05,
   size=[120,120], borderWidth=1.0,
   fillColor='black', borderColor='white',
   color='white', colorSpace='rgb',
   opacity=None,
   bold=True, italic=False,
   padding=None,
   anchor='center',
   name='button2_6'
)
button2_6.buttonClock = core.Clock()
button3_1 = visual.ButtonStim(win,
   text=None, font='Arvo',
   pos=[-300,-60],units='pix',
```

```
    letterHeight=0.05,
    size=[120,120], borderWidth=1.0,
    fillColor='black', borderColor='white',
    color='white', colorSpace='rgb',
    opacity=None,
    bold=True, italic=False,
    padding=None,
    anchor='center',
    name='button3_1'
)
button3_1.buttonClock = core.Clock()
button3_2 = visual.ButtonStim(win,
    text=None, font='Arvo',
    pos=[-180,-60],units='pix',
    letterHeight=0.05,
    size=[120,120], borderWidth=1.0,
    fillColor='black', borderColor='white',
    color='white', colorSpace='rgb',
    opacity=None,
    bold=True, italic=False,
    padding=None,
    anchor='center',
    name='button3_2'
)
button3_2.buttonClock = core.Clock()
button3_3 = visual.ButtonStim(win,
    text=None, font='Arvo',
    pos=[-60,-60],units='pix',
    letterHeight=0.05,
    size=[120,120], borderWidth=1.0,
    fillColor='black', borderColor='white',
    color='white', colorSpace='rgb',
    opacity=None,
    bold=True, italic=False,
    padding=None,
    anchor='center',
    name='button3_3'
)
button3_3.buttonClock = core.Clock()
button3_4 = visual.ButtonStim(win,
    text=None, font='Arvo',
    pos=[60,-60],units='pix',
    letterHeight=0.05,
    size=[120,120], borderWidth=1.0,
    fillColor='black', borderColor='white',
    color='white', colorSpace='rgb',
    opacity=None,
    bold=True, italic=False,
    padding=None,
    anchor='center',
    name='button3_4'
)
button3_4.buttonClock = core.Clock()
button3_5 = visual.ButtonStim(win,
    text=None, font='Arvo',
    pos=[180,-60],units='pix',
    letterHeight=0.05,
    size=[120,120], borderWidth=1.0,
    fillColor='black', borderColor='white',
```

```
    color='white', colorSpace='rgb',
    opacity=None,
    bold=True, italic=False,
    padding=None,
    anchor='center',
    name='button3_5'
)
button3_5.buttonClock = core.Clock()
button3_6 = visual.ButtonStim(win,
    text=None, font='Arvo',
    pos=[300,-60],units='pix',
    letterHeight=0.05,
    size=[120,120], borderWidth=1.0,
    fillColor='black', borderColor='white',
    color='white', colorSpace='rgb',
    opacity=None,
    bold=True, italic=False,
    padding=None,
    anchor='center',
    name='button3_6'
)
button3_6.buttonClock = core.Clock()
button4_1 = visual.ButtonStim(win,
    text=None, font='Arvo',
    pos=[-300,-180],units='pix',
    letterHeight=0.05,
    size=[120,120], borderWidth=1.0,
    fillColor='black', borderColor='white',
    color='white', colorSpace='rgb',
    opacity=None,
    bold=True, italic=False,
    padding=None,
    anchor='center',
    name='button4_1'
)
button4_1.buttonClock = core.Clock()
button4_2 = visual.ButtonStim(win,
    text=None, font='Arvo',
    pos=[-180,-180],units='pix',
    letterHeight=0.05,
    size=[120,120], borderWidth=1.0,
    fillColor='black', borderColor='white',
    color='white', colorSpace='rgb',
    opacity=None,
    bold=True, italic=False,
    padding=None,
    anchor='center',
    name='button4_2'
)
button4_2.buttonClock = core.Clock()
button4_3 = visual.ButtonStim(win,
    text=None, font='Arvo',
    pos=[-60,-180],units='pix',
    letterHeight=0.05,
    size=[120,120], borderWidth=1.0,
    fillColor='black', borderColor='white',
    color='white', colorSpace='rgb',
    opacity=None,
    bold=True, italic=False,
```

```
    padding=None,
    anchor='center',
    name='button4_3'
)
button4_3.buttonClock = core.Clock()
button4_4 = visual.ButtonStim(win,
    text=None, font='Arvo',
    pos=[60,-180],units='pix',
    letterHeight=0.05,
    size=[120,120], borderWidth=1.0,
    fillColor='black', borderColor='white',
    color='white', colorSpace='rgb',
    opacity=None,
    bold=True, italic=False,
    padding=None,
    anchor='center',
    name='button4_4'
)
button4_4.buttonClock = core.Clock()
button4_5 = visual.ButtonStim(win,
    text=None, font='Arvo',
    pos=[180,-180],units='pix',
    letterHeight=0.05,
    size=[120,120], borderWidth=1.0,
    fillColor='black', borderColor='white',
    color='white', colorSpace='rgb',
    opacity=None,
    bold=True, italic=False,
    padding=None,
    anchor='center',
    name='button4_5'
)
button4_5.buttonClock = core.Clock()
button4_6 = visual.ButtonStim(win,
    text=None, font='Arvo',
    pos=[300,-180],units='pix',
    letterHeight=0.05,
    size=[120,120], borderWidth=1.0,
    fillColor='black', borderColor='white',
    color='white', colorSpace='rgb',
    opacity=None,
    bold=True, italic=False,
    padding=None,
    anchor='center',
    name='button4_6'
)
button4_6.buttonClock = core.Clock()
ISI_3 = clock.StaticPeriod(win=win, screenHz=expInfo['frameRate'], name='ISI_3')

# Initialize components for Routine "trials_init"
trials_initClock = core.Clock()
instructions_4 = visual.TextStim(win=win, name='instructions_4',
    text="Κρατήστε κλειστά τα μάτια σας καθ' όλη τη διάρκεια της διαδικασίας.\n\n-Παραμένετε σε
ηρεμία μέχρι να ακούσετε ήχο.\n-Θα ακούσετε τη θέση στην οποία πρέπει να οδηγήσετε τον
δείκτη του κυρίαρχου χεριού.\n-Μόλις ακούστε ξανά ήχο, ξεκινήστε την κίνησή σας. Αγγίξτε την
επιθυμητή θέση και περιμένετε.\n-Μόλις ακούσετε ξανά ήχο, επαναφέρετε το χέρι σας στην
αρχική θέση και παραμείνετε σε ηρεμία μέχρι τον επόμενο ήχο.\n\nΠατήστε space όταν είστε
έτοιμοι.",
    font='Open Sans',
```

```
    pos=(0, 0), height=0.05, wrapWidth=None, ori=0.0,
    color='white', colorSpace='rgb', opacity=None,
    languageStyle='LTR',
    depth=0.0);
key_resp_4 = keyboard.Keyboard()

# Initialize components for Routine "go"
goClock = core.Clock()
ready_cue = sound.Sound('A', secs=1.0, stereo=True, hamming=True,
    name='ready_cue')
ready_cue.setVolume(1.0)
cue = sound.Sound('A', secs=1.5, stereo=True, hamming=True,
    name='cue')
cue.setVolume(1.0)
go_cue = sound.Sound('A', secs=1.0, stereo=True, hamming=True,
    name='go_cue')
go_cue.setVolume(1.0)
touchgrid = event.Mouse(visible = False, newPos = None, win=win)
x, y = [None, None]
touchgrid.mouseClock = core.Clock()
goblock1_1 = visual.Rect(
    win=win, name='goblock1_1',units='pix',
    width=(120, 120)[0], height=(120, 120)[1],
    ori=0.0, pos=(-300, 180),
    lineWidth=1.0,     colorSpace='rgb',  lineColor='white', fillColor='black',
    opacity=None, depth=-4.0, interpolate=True)
goblock1_2 = visual.Rect(
    win=win, name='goblock1_2',units='pix',
    width=(120, 120)[0], height=(120, 120)[1],
    ori=0.0, pos=(-180, 180),
    lineWidth=1.0,     colorSpace='rgb',  lineColor='white', fillColor='black',
    opacity=None, depth=-5.0, interpolate=True)
goblock1_3 = visual.Rect(
    win=win, name='goblock1_3',units='pix',
    width=(120, 120)[0], height=(120, 120)[1],
    ori=0.0, pos=(-60, 180),
    lineWidth=1.0,     colorSpace='rgb',  lineColor='white', fillColor='black',
    opacity=None, depth=-6.0, interpolate=True)
goblock1_4 = visual.Rect(
    win=win, name='goblock1_4',units='pix',
    width=(120, 120)[0], height=(120, 120)[1],
    ori=0.0, pos=(60, 180),
    lineWidth=1.0,     colorSpace='rgb',  lineColor='white', fillColor='black',
    opacity=None, depth=-7.0, interpolate=True)
goblock1_5 = visual.Rect(
    win=win, name='goblock1_5',units='pix',
    width=(120, 120)[0], height=(120, 120)[1],
    ori=0.0, pos=(180, 180),
    lineWidth=1.0,     colorSpace='rgb',  lineColor='white', fillColor='black',
    opacity=None, depth=-8.0, interpolate=True)
goblock1_6 = visual.Rect(
    win=win, name='goblock1_6',units='pix',
    width=(120, 120)[0], height=(120, 120)[1],
    ori=0.0, pos=(300, 180),
    lineWidth=1.0,     colorSpace='rgb',  lineColor='white', fillColor='black',
    opacity=None, depth=-9.0, interpolate=True)
goblock2_1 = visual.Rect(
    win=win, name='goblock2_1',units='pix',
    width=(120, 120)[0], height=(120, 120)[1],
```

```
    ori=0.0, pos=(-300, 60),
    lineWidth=1.0,     colorSpace='rgb',  lineColor='white', fillColor='black',
    opacity=None, depth=-10.0, interpolate=True)
goblock2_2 = visual.Rect(
    win=win, name='goblock2_2',units='pix',
    width=(120, 120)[0], height=(120, 120)[1],
    ori=0.0, pos=(-180, 60),
    lineWidth=1.0,     colorSpace='rgb',  lineColor='white', fillColor='black',
    opacity=None, depth=-11.0, interpolate=True)
goblock2_3 = visual.Rect(
    win=win, name='goblock2_3',units='pix',
    width=(120, 120)[0], height=(120, 120)[1],
    ori=0.0, pos=(-60, 60),
    lineWidth=1.0,     colorSpace='rgb',  lineColor='white', fillColor='black',
    opacity=None, depth=-12.0, interpolate=True)
goblock2_4 = visual.Rect(
    win=win, name='goblock2_4',units='pix',
    width=(120, 120)[0], height=(120, 120)[1],
    ori=0.0, pos=(60, 60),
    lineWidth=1.0,     colorSpace='rgb',  lineColor='white', fillColor='black',
    opacity=None, depth=-13.0, interpolate=True)
goblock2_5 = visual.Rect(
    win=win, name='goblock2_5',units='pix',
    width=(120, 120)[0], height=(120, 120)[1],
    ori=0.0, pos=(180, 60),
    lineWidth=1.0,     colorSpace='rgb',  lineColor='white', fillColor='black',
    opacity=None, depth=-14.0, interpolate=True)
goblock2_6 = visual.Rect(
    win=win, name='goblock2_6',units='pix',
    width=(120, 120)[0], height=(120, 120)[1],
    ori=0.0, pos=(300, 60),
    lineWidth=1.0,     colorSpace='rgb',  lineColor='white', fillColor='black',
    opacity=None, depth=-15.0, interpolate=True)
goblock3_1 = visual.Rect(
    win=win, name='goblock3_1',units='pix',
    width=(120, 120)[0], height=(120, 120)[1],
    ori=0.0, pos=(-300, -60),
    lineWidth=1.0,     colorSpace='rgb',  lineColor='white', fillColor='black',
    opacity=None, depth=-16.0, interpolate=True)
goblock3_2 = visual.Rect(
    win=win, name='goblock3_2',units='pix',
    width=(120, 120)[0], height=(120, 120)[1],
    ori=0.0, pos=(-180, -60),
    lineWidth=1.0,     colorSpace='rgb',  lineColor='white', fillColor='black',
    opacity=None, depth=-17.0, interpolate=True)
goblock3_3 = visual.Rect(
    win=win, name='goblock3_3',units='pix',
    width=(120, 120)[0], height=(120, 120)[1],
    ori=0.0, pos=(-60, -60),
    lineWidth=1.0,     colorSpace='rgb',  lineColor='white', fillColor='black',
    opacity=None, depth=-18.0, interpolate=True)
goblock3_4 = visual.Rect(
    win=win, name='goblock3_4',units='pix',
    width=(120, 120)[0], height=(120, 120)[1],
    ori=0.0, pos=(60, -60),
    lineWidth=1.0,     colorSpace='rgb',  lineColor='white', fillColor='black',
    opacity=None, depth=-19.0, interpolate=True)
goblock3_5 = visual.Rect(
    win=win, name='goblock3_5',units='pix',
```

```
    width=(120, 120)[0], height=(120, 120)[1],
    ori=0.0, pos=(180, -60),
    lineWidth=1.0,     colorSpace='rgb',  lineColor='white', fillColor='black',
    opacity=None, depth=-20.0, interpolate=True)
goblock3_6 = visual.Rect(
    win=win, name='goblock3_6',units='pix',
    width=(120, 120)[0], height=(120, 120)[1],
    ori=0.0, pos=(300, -60),
    lineWidth=1.0,     colorSpace='rgb',  lineColor='white', fillColor='black',
    opacity=None, depth=-21.0, interpolate=True)
goblock4_1 = visual.Rect(
    win=win, name='goblock4_1',units='pix',
    width=(120, 120)[0], height=(120, 120)[1],
    ori=0.0, pos=(-300, -180),
    lineWidth=1.0,     colorSpace='rgb',  lineColor='white', fillColor='black',
    opacity=None, depth=-22.0, interpolate=True)
goblock4_2 = visual.Rect(
    win=win, name='goblock4_2',units='pix',
    width=(120, 120)[0], height=(120, 120)[1],
    ori=0.0, pos=(-180, -180),
    lineWidth=1.0,     colorSpace='rgb',  lineColor='white', fillColor='black',
    opacity=None, depth=-23.0, interpolate=True)
goblock4_3 = visual.Rect(
    win=win, name='goblock4_3',units='pix',
    width=(120, 120)[0], height=(120, 120)[1],
    ori=0.0, pos=(-60, -180),
    lineWidth=1.0,     colorSpace='rgb',  lineColor='white', fillColor='black',
    opacity=None, depth=-24.0, interpolate=True)
goblock4_4 = visual.Rect(
    win=win, name='goblock4_4',units='pix',
    width=(120, 120)[0], height=(120, 120)[1],
    ori=0.0, pos=(60, -180),
    lineWidth=1.0,     colorSpace='rgb',  lineColor='white', fillColor='black',
    opacity=None, depth=-25.0, interpolate=True)
goblock4_5 = visual.Rect(
    win=win, name='goblock4_5',units='pix',
    width=(120, 120)[0], height=(120, 120)[1],
    ori=0.0, pos=(180, -180),
    lineWidth=1.0,     colorSpace='rgb',  lineColor='white', fillColor='black',
    opacity=None, depth=-26.0, interpolate=True)
goblock4_6 = visual.Rect(
    win=win, name='goblock4_6',units='pix',
    width=(120, 120)[0], height=(120, 120)[1],
    ori=0.0, pos=(300, -180),
    lineWidth=1.0,     colorSpace='rgb',  lineColor='white', fillColor='black',
    opacity=None, depth=-27.0, interpolate=True)

# Initialize components for Routine "back"
backClock = core.Clock()
return_cue = sound.Sound('A', secs=1.0, stereo=True, hamming=True,
    name='return_cue')
return_cue.setVolume(1.0)
trial_end = keyboard.Keyboard()
backgrid1_1 = visual.ButtonStim(win,
    text=None, font='Arvo',
    pos=[-300,180],units='pix',
    letterHeight=0.05,
    size=[120,120], borderWidth=1.0,
    fillColor='black', borderColor='white',
```

```
    color='white', colorSpace='rgb',
    opacity=None,
    bold=True, italic=False,
    padding=None,
    anchor='center',
    name='backgrid1_1'
)
backgrid1_1.buttonClock = core.Clock()
backgrid1_2 = visual.ButtonStim(win,
    text=None, font='Arvo',
    pos=[-180,180],units='pix',
    letterHeight=0.05,
    size=[120,120], borderWidth=1.0,
    fillColor='black', borderColor='white',
    color='white', colorSpace='rgb',
    opacity=None,
    bold=True, italic=False,
    padding=None,
    anchor='center',
    name='backgrid1_2'
)
backgrid1_2.buttonClock = core.Clock()
backgrid1_3 = visual.ButtonStim(win,
    text=None, font='Arvo',
    pos=[-60,180],units='pix',
    letterHeight=0.05,
    size=[120,120], borderWidth=1.0,
    fillColor='black', borderColor='white',
    color='white', colorSpace='rgb',
    opacity=None,
    bold=True, italic=False,
    padding=None,
    anchor='center',
    name='backgrid1_3'
)
backgrid1_3.buttonClock = core.Clock()
backgrid1_4 = visual.ButtonStim(win,
    text=None, font='Arvo',
    pos=[60,180],units='pix',
    letterHeight=0.05,
    size=[120,120], borderWidth=1.0,
    fillColor='black', borderColor='white',
    color='white', colorSpace='rgb',
    opacity=None,
    bold=True, italic=False,
    padding=None,
    anchor='center',
    name='backgrid1_4'
)
backgrid1_4.buttonClock = core.Clock()
backgrid1_5 = visual.ButtonStim(win,
    text='xaxaxa', font='Arvo',
    pos=[180,180],units='pix',
    letterHeight=0.05,
    size=[120,120], borderWidth=1.0,
    fillColor='black', borderColor='white',
    color='white', colorSpace='rgb',
    opacity=None,
    bold=True, italic=False,
```

```
    padding=None,
    anchor='center',
    name='backgrid1_5'
)
backgrid1_5.buttonClock = core.Clock()
backgrid1_6 = visual.ButtonStim(win,
    text='xaxaxa', font='Arvo',
    pos=[300,180],units='pix',
    letterHeight=0.05,
    size=[120,120], borderWidth=1.0,
    fillColor='black', borderColor='white',
    color='white', colorSpace='rgb',
    opacity=None,
    bold=True, italic=False,
    padding=None,
    anchor='center',
    name='backgrid1_6'
)
backgrid1_6.buttonClock = core.Clock()
backgrid2_1 = visual.ButtonStim(win,
    text=None, font='Arvo',
    pos=[-300,60],units='pix',
    letterHeight=0.05,
    size=[120,120], borderWidth=1.0,
    fillColor='black', borderColor='white',
    color='white', colorSpace='rgb',
    opacity=None,
    bold=True, italic=False,
    padding=None,
    anchor='center',
    name='backgrid2_1'
)
backgrid2_1.buttonClock = core.Clock()
backgrid2_2 = visual.ButtonStim(win,
    text=None, font='Arvo',
    pos=[-180,60],units='pix',
    letterHeight=0.05,
    size=[120,120], borderWidth=1.0,
    fillColor='black', borderColor='white',
    color='white', colorSpace='rgb',
    opacity=None,
    bold=True, italic=False,
    padding=None,
    anchor='center',
    name='backgrid2_2'
)
backgrid2_2.buttonClock = core.Clock()
backgrid2_3 = visual.ButtonStim(win,
    text=None, font='Arvo',
    pos=[-60,60],units='pix',
    letterHeight=0.05,
    size=[120,120], borderWidth=1.0,
    fillColor='black', borderColor='white',
    color='white', colorSpace='rgb',
    opacity=None,
    bold=True, italic=False,
    padding=None,
    anchor='center',
    name='backgrid2_3'
```

```
)
backgrid2_3.buttonClock = core.Clock()
backgrid2_4 = visual.ButtonStim(win,
   text=None, font='Arvo',
   pos=[60,60],units='pix',
   letterHeight=0.05,
   size=[120,120], borderWidth=1.0,
   fillColor='black', borderColor='white',
   color='white', colorSpace='rgb',
   opacity=None,
   bold=True, italic=False,
   padding=None,
   anchor='center',
   name='backgrid2_4'
)
backgrid2_4.buttonClock = core.Clock()
backgrid2_5 = visual.ButtonStim(win,
   text=None, font='Arvo',
   pos=[180,60],units='pix',
   letterHeight=0.05,
   size=[120,120], borderWidth=1.0,
   fillColor='black', borderColor='white',
   color='white', colorSpace='rgb',
   opacity=None,
   bold=True, italic=False,
   padding=None,
   anchor='center',
   name='backgrid2_5'
)
backgrid2_5.buttonClock = core.Clock()
backgrid2_6 = visual.ButtonStim(win,
   text=None, font='Arvo',
   pos=[300,60],units='pix',
   letterHeight=0.05,
   size=[120,120], borderWidth=1.0,
   fillColor='black', borderColor='white',
   color='white', colorSpace='rgb',
   opacity=None,
   bold=True, italic=False,
   padding=None,
   anchor='center',
   name='backgrid2_6'
)
backgrid2_6.buttonClock = core.Clock()
backgrid3_1 = visual.ButtonStim(win,
   text=None, font='Arvo',
   pos=[-300,-60],units='pix',
   letterHeight=0.05,
   size=[120,120], borderWidth=1.0,
   fillColor='black', borderColor='white',
   color='white', colorSpace='rgb',
   opacity=None,
   bold=True, italic=False,
   padding=None,
   anchor='center',
   name='backgrid3_1'
)
backgrid3_1.buttonClock = core.Clock()
backgrid3_2 = visual.ButtonStim(win,
```

```
    text=None, font='Arvo',
    pos=[-180,-60],units='pix',
    letterHeight=0.05,
    size=[120,120], borderWidth=1.0,
    fillColor='black', borderColor='white',
    color='white', colorSpace='rgb',
    opacity=None,
    bold=True, italic=False,
    padding=None,
    anchor='center',
    name='backgrid3_2'
)
backgrid3_2.buttonClock = core.Clock()
backgrid3_3 = visual.ButtonStim(win,
    text=None, font='Arvo',
    pos=[-60,-60],units='pix',
    letterHeight=0.05,
    size=[120,120], borderWidth=1.0,
    fillColor='black', borderColor='white',
    color='white', colorSpace='rgb',
    opacity=None,
    bold=True, italic=False,
    padding=None,
    anchor='center',
    name='backgrid3_3'
)
backgrid3_3.buttonClock = core.Clock()
backgrid3_4 = visual.ButtonStim(win,
    text=None, font='Arvo',
    pos=[60,-60],units='pix',
    letterHeight=0.05,
    size=[120,120], borderWidth=1.0,
    fillColor='black', borderColor='white',
    color='white', colorSpace='rgb',
    opacity=None,
    bold=True, italic=False,
    padding=None,
    anchor='center',
    name='backgrid3_4'
)
backgrid3_4.buttonClock = core.Clock()
backgrid3_5 = visual.ButtonStim(win,
    text=None, font='Arvo',
    pos=[180,-60],units='pix',
    letterHeight=0.05,
    size=[120,120], borderWidth=1.0,
    fillColor='black', borderColor='white',
    color='white', colorSpace='rgb',
    opacity=None,
    bold=True, italic=False,
    padding=None,
    anchor='center',
    name='backgrid3_5'
)
backgrid3_5.buttonClock = core.Clock()
backgrid3_6 = visual.ButtonStim(win,
    text=None, font='Arvo',
    pos=[300,-60],units='pix',
    letterHeight=0.05,
```

```
    size=[120,120], borderWidth=1.0,
    fillColor='black', borderColor='white',
    color='white', colorSpace='rgb',
    opacity=None,
    bold=True, italic=False,
    padding=None,
    anchor='center',
    name='backgrid3_6'
)
backgrid3_6.buttonClock = core.Clock()
backgrid4_1 = visual.ButtonStim(win,
    text=None, font='Arvo',
    pos=[-300,-180],units='pix',
    letterHeight=0.05,
    size=[120,120], borderWidth=1.0,
    fillColor='black', borderColor='white',
    color='white', colorSpace='rgb',
    opacity=None,
    bold=True, italic=False,
    padding=None,
    anchor='center',
    name='backgrid4_1'
)
backgrid4_1.buttonClock = core.Clock()
backgrid4_2 = visual.ButtonStim(win,
    text=None, font='Arvo',
    pos=[-180,-180],units='pix',
    letterHeight=0.05,
    size=[120,120], borderWidth=1.0,
    fillColor='black', borderColor='white',
    color='white', colorSpace='rgb',
    opacity=None,
    bold=True, italic=False,
    padding=None,
    anchor='center',
    name='backgrid4_2'
)
backgrid4_2.buttonClock = core.Clock()
backgrid4_3 = visual.ButtonStim(win,
    text=None, font='Arvo',
    pos=[-60,-180],units='pix',
    letterHeight=0.05,
    size=[120,120], borderWidth=1.0,
    fillColor='black', borderColor='white',
    color='white', colorSpace='rgb',
    opacity=None,
    bold=True, italic=False,
    padding=None,
    anchor='center',
    name='backgrid4_3'
)
backgrid4_3.buttonClock = core.Clock()
backgrid4_4 = visual.ButtonStim(win,
    text=None, font='Arvo',
    pos=[60,-180],units='pix',
    letterHeight=0.05,
    size=[120,120], borderWidth=1.0,
    fillColor='black', borderColor='white',
    color='white', colorSpace='rgb',
```

```
    opacity=None,
    bold=True, italic=False,
    padding=None,
    anchor='center',
    name='backgrid4_4'
)
backgrid4_4.buttonClock = core.Clock()
backgrid4_5 = visual.ButtonStim(win,
    text=None, font='Arvo',
    pos=[180,-180],units='pix',
    letterHeight=0.05,
    size=[120,120], borderWidth=1.0,
    fillColor='black', borderColor='white',
    color='white', colorSpace='rgb',
    opacity=None,
    bold=True, italic=False,
    padding=None,
    anchor='center',
    name='backgrid4_5'
)
backgrid4_5.buttonClock = core.Clock()
backgrid4_6 = visual.ButtonStim(win,
    text=None, font='Arvo',
    pos=[300,-180],units='pix',
    letterHeight=0.05,
    size=[120,120], borderWidth=1.0,
    fillColor='black', borderColor='white',
    color='white', colorSpace='rgb',
    opacity=None,
    bold=True, italic=False,
    padding=None,
    anchor='center',
    name='backgrid4_6'
)
backgrid4_6.buttonClock = core.Clock()

# Create some handy timers
globalClock = core.Clock()  # to track the time since experiment started
routineTimer = core.CountdownTimer()  # to track time remaining of each (non-slip) routine
```

Ροή Πειράματος

```
# ------Prepare to start Routine "rest_init"-------
continueRoutine = True
# update component parameters for each repeat
key_resp.keys = []
key_resp.rt = []
_key_resp_allKeys = []
# keep track of which components have finished
rest_initComponents = [instructions, key_resp]
for thisComponent in rest_initComponents:
    thisComponent.tStart = None
    thisComponent.tStop = None
    thisComponent.tStartRefresh = None
    thisComponent.tStopRefresh = None
    if hasattr(thisComponent, 'status'):
```

```
    thisComponent.status = NOT_STARTED
# reset timers
t = 0
_timeToFirstFrame = win.getFutureFlipTime(clock="now")
rest_initClock.reset(-_timeToFirstFrame)  # t0 is time of first possible flip
frameN = -1

fileout = filename+"_custom.csv"
original_stdout = sys.stdout # Save a reference to the original standard output
with open(fileout,'w') as f:
  sys.stdout = f # Change the standard output to the file
  print("timestamp;label;trial;response;corrAns;pos_x;pos_y")

  # -------Run Routine "rest_init"-------
  while continueRoutine:
    # get current time
    t = rest_initClock.getTime()
    tThisFlip = win.getFutureFlipTime(clock=rest_initClock)
    tThisFlipGlobal = win.getFutureFlipTime(clock=None)
    frameN = frameN + 1  # number of completed frames (so 0 is the first frame)
    # update/draw components on each frame

    # *instructions* updates
    if instructions.status == NOT_STARTED and tThisFlip >= 0.0-frameTolerance:
      # keep track of start time/frame for later
      instructions.frameNStart = frameN  # exact frame index
      instructions.tStart = t  # local t and not account for scr refresh
      instructions.tStartRefresh = tThisFlipGlobal  # on global time
      win.timeOnFlip(instructions, 'tStartRefresh')  # time at next scr refresh
      instructions.setAutoDraw(True)

    # *key_resp* updates
    waitOnFlip = False
    if key_resp.status == NOT_STARTED and tThisFlip >= 0.0-frameTolerance:
      # keep track of start time/frame for later
      key_resp.frameNStart = frameN  # exact frame index
      key_resp.tStart = t  # local t and not account for scr refresh
      key_resp.tStartRefresh = tThisFlipGlobal  # on global time
      win.timeOnFlip(key_resp, 'tStartRefresh')  # time at next scr refresh
      key_resp.status = STARTED
      # keyboard checking is just starting
      waitOnFlip = True
      win.callOnFlip(key_resp.clock.reset)  # t=0 on next screen flip
      win.callOnFlip(key_resp.clearEvents, eventType='keyboard')  # clear events on next screen flip
    if key_resp.status == STARTED and not waitOnFlip:
      theseKeys = key_resp.getKeys(keyList=['space'], waitRelease=False)
      _key_resp_allKeys.extend(theseKeys)
      if len(_key_resp_allKeys):
        key_resp.keys = _key_resp_allKeys[-1].name  # just the last key pressed
        key_resp.rt = _key_resp_allKeys[-1].rt
        # a response ends the routine
        continueRoutine = False

    # check for quit (typically the Esc key)
    if endExpNow or defaultKeyboard.getKeys(keyList=["escape"]):
      core.quit()

    # check if all components have finished
    if not continueRoutine:  # a component has requested a forced-end of Routine
```

```
        break
    continueRoutine = False  # will revert to True if at least one component still running
    for thisComponent in rest_initComponents:
        if hasattr(thisComponent, "status") and thisComponent.status != FINISHED:
            continueRoutine = True
            break  # at least one component has not yet finished

    # refresh the screen
    if continueRoutine:  # don't flip if this routine is over or we'll get a blank screen
        win.flip()

# -------Ending Routine "rest_init"-------
for thisComponent in rest_initComponents:
    if hasattr(thisComponent, "setAutoDraw"):
        thisComponent.setAutoDraw(False)
# the Routine "rest_init" was not non-slip safe, so reset the non-slip timer
routineTimer.reset()

# ------Prepare to start Routine "rest"-------
continueRoutine = True
routineTimer.add(7.100000)
# update component parameters for each repeat
rest_go.setSound('Censor Beep Sound Effect.wav', secs=1.0, hamming=True)
rest_go.setVolume(1.0, log=False)
rest_stop.setSound('Censor Beep Sound Effect.wav', secs=1.0, hamming=True)
rest_stop.setVolume(1.0, log=False)
# keep track of which components have finished
restComponents = [rest_go, cross_rest, rest_stop, ISI]
for thisComponent in restComponents:
    thisComponent.tStart = None
    thisComponent.tStop = None
    thisComponent.tStartRefresh = None
    thisComponent.tStopRefresh = None
    if hasattr(thisComponent, 'status'):
        thisComponent.status = NOT_STARTED
# reset timers
t = 0
_timeToFirstFrame = win.getFutureFlipTime(clock="now")
restClock.reset(-_timeToFirstFrame)  # t0 is time of first possible flip
frameN = -1

# -------Run Routine "rest"-------
while continueRoutine and routineTimer.getTime() > 0:
    # get current time
    t = restClock.getTime()
    tThisFlip = win.getFutureFlipTime(clock=restClock)
    tThisFlipGlobal = win.getFutureFlipTime(clock=None)
    frameN = frameN + 1  # number of completed frames (so 0 is the first frame)
    # update/draw components on each frame
    # start/stop rest_go
    if rest_go.status == NOT_STARTED and tThisFlip >= 0-frameTolerance:
        # keep track of start time/frame for later
        rest_go.frameNStart = frameN  # exact frame index
        rest_go.tStart = t  # local t and not account for scr refresh
        rest_go.tStartRefresh = tThisFlipGlobal  # on global time
        rest_go.play(when=win)  # sync with win flip
    if rest_go.status == STARTED:
        # is it time to stop? (based on global clock, using actual start)
        if tThisFlipGlobal > rest_go.tStartRefresh + 1.0-frameTolerance:
```

```
        # keep track of stop time/frame for later
        rest_go.tStop = t  # not accounting for scr refresh
        rest_go.frameNStop = frameN  # exact frame index
        win.timeOnFlip(rest_go, 'tStopRefresh')  # time at next scr refresh
        rest_go.stop()

    # *cross_rest* updates
    if cross_rest.status == NOT_STARTED and tThisFlip >= 1-frameTolerance:
        # keep track of start time/frame for later
        cross_rest.frameNStart = frameN  # exact frame index
        cross_rest.tStart = t  # local t and not account for scr refresh
        cross_rest.tStartRefresh = tThisFlipGlobal  # on global time
        win.timeOnFlip(cross_rest, 'tStartRefresh')  # time at next scr refresh
        cross_rest.setAutoDraw(True)
    if cross_rest.status == STARTED:
        # is it time to stop? (based on global clock, using actual start)
        if tThisFlipGlobal > cross_rest.tStartRefresh + 5.0-frameTolerance:
            # keep track of stop time/frame for later
            cross_rest.tStop = t  # not accounting for scr refresh
            cross_rest.frameNStop = frameN  # exact frame index
            win.timeOnFlip(cross_rest, 'tStopRefresh')  # time at next scr refresh
            cross_rest.setAutoDraw(False)
    # start/stop rest_stop
    if rest_stop.status == NOT_STARTED and tThisFlip >= 6-frameTolerance:
        # keep track of start time/frame for later
        rest_stop.frameNStart = frameN  # exact frame index
        rest_stop.tStart = t  # local t and not account for scr refresh
        rest_stop.tStartRefresh = tThisFlipGlobal  # on global time
        rest_stop.play(when=win)  # sync with win flip
    if rest_stop.status == STARTED:
        # is it time to stop? (based on global clock, using actual start)
        if tThisFlipGlobal > rest_stop.tStartRefresh + 1.0-frameTolerance:
            # keep track of stop time/frame for later
            rest_stop.tStop = t  # not accounting for scr refresh
            rest_stop.frameNStop = frameN  # exact frame index
            win.timeOnFlip(rest_stop, 'tStopRefresh')  # time at next scr refresh
            rest_stop.stop()
    # *ISI* period
    if ISI.status == NOT_STARTED and t >= 7-frameTolerance:
        # keep track of start time/frame for later
        ISI.frameNStart = frameN  # exact frame index
        ISI.tStart = t  # local t and not account for scr refresh
        ISI.tStartRefresh = tThisFlipGlobal  # on global time
        win.timeOnFlip(ISI, 'tStartRefresh')  # time at next scr refresh
        ISI.start(0.1)
    elif ISI.status == STARTED:  # one frame should pass before updating params and completing
        ISI.complete()  # finish the static period
        ISI.tStop = ISI.tStart + 0.1  # record stop time

    # check for quit (typically the Esc key)
    if endExpNow or defaultKeyboard.getKeys(keyList=["escape"]):
        core.quit()

    # check if all components have finished
    if not continueRoutine:  # a component has requested a forced-end of Routine
        break
    continueRoutine = False  # will revert to True if at least one component still running
    for thisComponent in restComponents:
        if hasattr(thisComponent, "status") and thisComponent.status != FINISHED:
```

```
            continueRoutine = True
            break  # at least one component has not yet finished

        # refresh the screen
        if continueRoutine:  # don't flip if this routine is over or we'll get a blank screen
            win.flip()

    # -------Ending Routine "rest"-------
    for thisComponent in restComponents:
        if hasattr(thisComponent, "setAutoDraw"):
            thisComponent.setAutoDraw(False)
    rest_go.stop()  # ensure sound has stopped at end of routine
    thisExp.addData('rest_go.started', rest_go.tStartRefresh)
    thisExp.addData('rest_go.stopped', rest_go.tStopRefresh)
    rest_stop.stop()  # ensure sound has stopped at end of routine
    thisExp.addData('rest_stop.started', rest_stop.tStartRefresh)
    thisExp.addData('rest_stop.stopped', rest_stop.tStopRefresh)

    print(str(rest_go.tStartRefresh)+";"+"rest_go.started"+";-;-;-;-;-")
    print(str(rest_go.tStopRefresh)+";"+"rest_go.stopped"+";-;-;-;-;-")
    print(str(rest_stop.tStartRefresh)+";"+"rest_stop.started"+";-;-;-;-;-")
    print(str(rest_stop.tStopRefresh)+";"+"rest_stop.stopped"+";-;-;-;-;-")

    # ------Prepare to start Routine "movement_init"-------
    continueRoutine = True
    # update component parameters for each repeat
    key_resp_2.keys = []
    key_resp_2.rt = []
    _key_resp_2_allKeys = []
    # keep track of which components have finished
    movement_initComponents = [instructions_2, key_resp_2]
    for thisComponent in movement_initComponents:
        thisComponent.tStart = None
        thisComponent.tStop = None
        thisComponent.tStartRefresh = None
        thisComponent.tStopRefresh = None
        if hasattr(thisComponent, 'status'):
            thisComponent.status = NOT_STARTED
    # reset timers
    t = 0
    _timeToFirstFrame = win.getFutureFlipTime(clock="now")
    movement_initClock.reset(-_timeToFirstFrame)  # t0 is time of first possible flip
    frameN = -1

    # -------Run Routine "movement_init"-------
    while continueRoutine:
        # get current time
        t = movement_initClock.getTime()
        tThisFlip = win.getFutureFlipTime(clock=movement_initClock)
        tThisFlipGlobal = win.getFutureFlipTime(clock=None)
        frameN = frameN + 1  # number of completed frames (so 0 is the first frame)
        # update/draw components on each frame

        # *instructions_2* updates
        if instructions_2.status == NOT_STARTED and tThisFlip >= 0.0-frameTolerance:
            # keep track of start time/frame for later
            instructions_2.frameNStart = frameN  # exact frame index
            instructions_2.tStart = t  # local t and not account for scr refresh
            instructions_2.tStartRefresh = tThisFlipGlobal  # on global time
```

```
        win.timeOnFlip(instructions_2, 'tStartRefresh')  # time at next scr refresh
        instructions_2.setAutoDraw(True)

    # *key_resp_2* updates
    waitOnFlip = False
    if key_resp_2.status == NOT_STARTED and tThisFlip >= 0.0-frameTolerance:
        # keep track of start time/frame for later
        key_resp_2.frameNStart = frameN  # exact frame index
        key_resp_2.tStart = t  # local t and not account for scr refresh
        key_resp_2.tStartRefresh = tThisFlipGlobal  # on global time
        win.timeOnFlip(key_resp_2, 'tStartRefresh')  # time at next scr refresh
        key_resp_2.status = STARTED
        # keyboard checking is just starting
        waitOnFlip = True
        win.callOnFlip(key_resp_2.clock.reset)  # t=0 on next screen flip
        win.callOnFlip(key_resp_2.clearEvents, eventType='keyboard')  # clear events on next screen
flip
    if key_resp_2.status == STARTED and not waitOnFlip:
        theseKeys = key_resp_2.getKeys(keyList=['space'], waitRelease=False)
        _key_resp_2_allKeys.extend(theseKeys)
        if len(_key_resp_2_allKeys):
            key_resp_2.keys = _key_resp_2_allKeys[-1].name  # just the last key pressed
            key_resp_2.rt = _key_resp_2_allKeys[-1].rt
            # a response ends the routine
            continueRoutine = False

    # check for quit (typically the Esc key)
    if endExpNow or defaultKeyboard.getKeys(keyList=["escape"]):
        core.quit()

    # check if all components have finished
    if not continueRoutine:  # a component has requested a forced-end of Routine
        break
    continueRoutine = False  # will revert to True if at least one component still running
    for thisComponent in movement_initComponents:
        if hasattr(thisComponent, "status") and thisComponent.status != FINISHED:
            continueRoutine = True
            break  # at least one component has not yet finished

    # refresh the screen
    if continueRoutine:  # don't flip if this routine is over or we'll get a blank screen
        win.flip()

# -------Ending Routine "movement_init"-------
for thisComponent in movement_initComponents:
    if hasattr(thisComponent, "setAutoDraw"):
        thisComponent.setAutoDraw(False)
# the Routine "movement_init" was not non-slip safe, so reset the non-slip timer
routineTimer.reset()

# ------Prepare to start Routine "movement"-------
continueRoutine = True
routineTimer.add(7.100000)
# update component parameters for each repeat
move_go.setSound('Censor Beep Sound Effect.wav', secs=1.0, hamming=True)
move_go.setVolume(1.0, log=False)
move_stop.setSound('Censor Beep Sound Effect.wav', secs=1.0, hamming=True)
move_stop.setVolume(1.0, log=False)
# keep track of which components have finished
```

```
movementComponents = [move_go, move_stop, ISI_2]
for thisComponent in movementComponents:
    thisComponent.tStart = None
    thisComponent.tStop = None
    thisComponent.tStartRefresh = None
    thisComponent.tStopRefresh = None
    if hasattr(thisComponent, 'status'):
        thisComponent.status = NOT_STARTED
# reset timers
t = 0
_timeToFirstFrame = win.getFutureFlipTime(clock="now")
movementClock.reset(-_timeToFirstFrame)  # t0 is time of first possible flip
frameN = -1

# -------Run Routine "movement"-------
while continueRoutine and routineTimer.getTime() > 0:
    # get current time
    t = movementClock.getTime()
    tThisFlip = win.getFutureFlipTime(clock=movementClock)
    tThisFlipGlobal = win.getFutureFlipTime(clock=None)
    frameN = frameN + 1  # number of completed frames (so 0 is the first frame)
    # update/draw components on each frame
    # start/stop move_go
    if move_go.status == NOT_STARTED and tThisFlip >= 0-frameTolerance:
        # keep track of start time/frame for later
        move_go.frameNStart = frameN  # exact frame index
        move_go.tStart = t  # local t and not account for scr refresh
        move_go.tStartRefresh = tThisFlipGlobal  # on global time
        move_go.play(when=win)  # sync with win flip
    if move_go.status == STARTED:
        # is it time to stop? (based on global clock, using actual start)
        if tThisFlipGlobal > move_go.tStartRefresh + 1.0-frameTolerance:
            # keep track of stop time/frame for later
            move_go.tStop = t  # not accounting for scr refresh
            move_go.frameNStop = frameN  # exact frame index
            win.timeOnFlip(move_go, 'tStopRefresh')  # time at next scr refresh
            move_go.stop()
    # start/stop move_stop
    if move_stop.status == NOT_STARTED and tThisFlip >= 6-frameTolerance:
        # keep track of start time/frame for later
        move_stop.frameNStart = frameN  # exact frame index
        move_stop.tStart = t  # local t and not account for scr refresh
        move_stop.tStartRefresh = tThisFlipGlobal  # on global time
        move_stop.play(when=win)  # sync with win flip
    if move_stop.status == STARTED:
        # is it time to stop? (based on global clock, using actual start)
        if tThisFlipGlobal > move_stop.tStartRefresh + 1.0-frameTolerance:
            # keep track of stop time/frame for later
            move_stop.tStop = t  # not accounting for scr refresh
            move_stop.frameNStop = frameN  # exact frame index
            win.timeOnFlip(move_stop, 'tStopRefresh')  # time at next scr refresh
            move_stop.stop()
    # *ISI_2* period
    if ISI_2.status == NOT_STARTED and t >= 7-frameTolerance:
        # keep track of start time/frame for later
        ISI_2.frameNStart = frameN  # exact frame index
        ISI_2.tStart = t  # local t and not account for scr refresh
        ISI_2.tStartRefresh = tThisFlipGlobal  # on global time
        win.timeOnFlip(ISI_2, 'tStartRefresh')  # time at next scr refresh
```

```
        ISI_2.start(0.1)
    elif ISI_2.status == STARTED:  # one frame should pass before updating params and completing
        ISI_2.complete()  # finish the static period
        ISI_2.tStop = ISI_2.tStart + 0.1  # record stop time

    # check for quit (typically the Esc key)
    if endExpNow or defaultKeyboard.getKeys(keyList=["escape"]):
        core.quit()

    # check if all components have finished
    if not continueRoutine:  # a component has requested a forced-end of Routine
        break
    continueRoutine = False  # will revert to True if at least one component still running
    for thisComponent in movementComponents:
        if hasattr(thisComponent, "status") and thisComponent.status != FINISHED:
            continueRoutine = True
            break  # at least one component has not yet finished

    # refresh the screen
    if continueRoutine:  # don't flip if this routine is over or we'll get a blank screen
        win.flip()

# -------Ending Routine "movement"-------
for thisComponent in movementComponents:
    if hasattr(thisComponent, "setAutoDraw"):
        thisComponent.setAutoDraw(False)
move_go.stop()  # ensure sound has stopped at end of routine
thisExp.addData('move_go.started', move_go.tStartRefresh)
thisExp.addData('move_go.stopped', move_go.tStopRefresh)
move_stop.stop()  # ensure sound has stopped at end of routine
thisExp.addData('move_stop.started', move_stop.tStartRefresh)
thisExp.addData('move_stop.stopped', move_stop.tStopRefresh)

print(str(move_go.tStartRefresh)+";"+"move_go.started"+";-;-;-;-;-")
print(str(move_go.tStopRefresh)+";"+"move_go.stopped"+";-;-;-;-;-")
print(str(move_stop.tStartRefresh)+";"+"move_stop.started"+";-;-;-;-;-")
print(str(move_stop.tStopRefresh)+";"+"move_stop.stopped"+";-;-;-;-;-")

# ------Prepare to start Routine "encoding_init"-------
continueRoutine = True
# update component parameters for each repeat
key_resp_3.keys = []
key_resp_3.rt = []
_key_resp_3_allKeys = []
# keep track of which components have finished
encoding_initComponents = [instructions_3, key_resp_3]
for thisComponent in encoding_initComponents:
    thisComponent.tStart = None
    thisComponent.tStop = None
    thisComponent.tStartRefresh = None
    thisComponent.tStopRefresh = None
    if hasattr(thisComponent, 'status'):
        thisComponent.status = NOT_STARTED
# reset timers
t = 0
_timeToFirstFrame = win.getFutureFlipTime(clock="now")
encoding_initClock.reset(-_timeToFirstFrame)  # t0 is time of first possible flip
frameN = -1
```

```
    # -------Run Routine "encoding_init"-------
    while continueRoutine:
        # get current time
        t = encoding_initClock.getTime()
        tThisFlip = win.getFutureFlipTime(clock=encoding_initClock)
        tThisFlipGlobal = win.getFutureFlipTime(clock=None)
        frameN = frameN + 1  # number of completed frames (so 0 is the first frame)
        # update/draw components on each frame

        # *instructions_3* updates
        if instructions_3.status == NOT_STARTED and tThisFlip >= 0.0-frameTolerance:
            # keep track of start time/frame for later
            instructions_3.frameNStart = frameN  # exact frame index
            instructions_3.tStart = t  # local t and not account for scr refresh
            instructions_3.tStartRefresh = tThisFlipGlobal  # on global time
            win.timeOnFlip(instructions_3, 'tStartRefresh')  # time at next scr refresh
            instructions_3.setAutoDraw(True)

        # *key_resp_3* updates
        waitOnFlip = False
        if key_resp_3.status == NOT_STARTED and tThisFlip >= 0.0-frameTolerance:
            # keep track of start time/frame for later
            key_resp_3.frameNStart = frameN  # exact frame index
            key_resp_3.tStart = t  # local t and not account for scr refresh
            key_resp_3.tStartRefresh = tThisFlipGlobal  # on global time
            win.timeOnFlip(key_resp_3, 'tStartRefresh')  # time at next scr refresh
            key_resp_3.status = STARTED
            # keyboard checking is just starting
            waitOnFlip = True
            win.callOnFlip(key_resp_3.clock.reset)  # t=0 on next screen flip
            win.callOnFlip(key_resp_3.clearEvents, eventType='keyboard')  # clear events on next screen
flip
        if key_resp_3.status == STARTED and not waitOnFlip:
            theseKeys = key_resp_3.getKeys(keyList=['space'], waitRelease=False)
            _key_resp_3_allKeys.extend(theseKeys)
            if len(_key_resp_3_allKeys):
                key_resp_3.keys = _key_resp_3_allKeys[-1].name  # just the last key pressed
                key_resp_3.rt = _key_resp_3_allKeys[-1].rt
                # a response ends the routine
                continueRoutine = False

        # check for quit (typically the Esc key)
        if endExpNow or defaultKeyboard.getKeys(keyList=["escape"]):
            core.quit()

        # check if all components have finished
        if not continueRoutine:  # a component has requested a forced-end of Routine
            break
        continueRoutine = False  # will revert to True if at least one component still running
        for thisComponent in encoding_initComponents:
            if hasattr(thisComponent, "status") and thisComponent.status != FINISHED:
                continueRoutine = True
                break  # at least one component has not yet finished

        # refresh the screen
        if continueRoutine:  # don't flip if this routine is over or we'll get a blank screen
            win.flip()

    # -------Ending Routine "encoding_init"-------
```

```
    for thisComponent in encoding_initComponents:
        if hasattr(thisComponent, "setAutoDraw"):
            thisComponent.setAutoDraw(False)
    # the Routine "encoding_init" was not non-slip safe, so reset the non-slip timer
    routineTimer.reset()

    # ------Prepare to start Routine "encoding"-------
    continueRoutine = True
    routineTimer.add(32.100000)
    # update component parameters for each repeat
    encoding_go.setSound('Censor Beep Sound Effect.wav', secs=1.0, hamming=True)
    encoding_go.setVolume(1.0, log=False)
    encoding_stop.setSound('Censor Beep Sound Effect.wav', secs=1, hamming=True)
    encoding_stop.setVolume(1.0, log=False)
    # keep track of which components have finished
    encodingComponents = [encoding_go, encoding_stop, button1_1, button1_2, button1_3,
button1_4, button1_5, button1_6, button2_1, button2_2, button2_3, button2_4, button2_5,
button2_6, button3_1, button3_2, button3_3, button3_4, button3_5, button3_6, button4_1,
button4_2, button4_3, button4_4, button4_5, button4_6, ISI_3]
    for thisComponent in encodingComponents:
        thisComponent.tStart = None
        thisComponent.tStop = None
        thisComponent.tStartRefresh = None
        thisComponent.tStopRefresh = None
        if hasattr(thisComponent, 'status'):
            thisComponent.status = NOT_STARTED
    # reset timers
    t = 0
    _timeToFirstFrame = win.getFutureFlipTime(clock="now")
    encodingClock.reset(-_timeToFirstFrame)  # t0 is time of first possible flip
    frameN = -1

    # -------Run Routine "encoding"-------
    while continueRoutine and routineTimer.getTime() > 0:
        # get current time
        t = encodingClock.getTime()
        tThisFlip = win.getFutureFlipTime(clock=encodingClock)
        tThisFlipGlobal = win.getFutureFlipTime(clock=None)
        frameN = frameN + 1  # number of completed frames (so 0 is the first frame)
        # update/draw components on each frame
        # start/stop encoding_go
        if encoding_go.status == NOT_STARTED and tThisFlip >= 0-frameTolerance:
            # keep track of start time/frame for later
            encoding_go.frameNStart = frameN  # exact frame index
            encoding_go.tStart = t  # local t and not account for scr refresh
            encoding_go.tStartRefresh = tThisFlipGlobal  # on global time
            encoding_go.play(when=win)  # sync with win flip
        if encoding_go.status == STARTED:
            # is it time to stop? (based on global clock, using actual start)
            if tThisFlipGlobal > encoding_go.tStartRefresh + 1.0-frameTolerance:
                # keep track of stop time/frame for later
                encoding_go.tStop = t  # not accounting for scr refresh
                encoding_go.frameNStop = frameN  # exact frame index
                win.timeOnFlip(encoding_go, 'tStopRefresh')  # time at next scr refresh
                encoding_go.stop()
        # start/stop encoding_stop
        if encoding_stop.status == NOT_STARTED and tThisFlip >= 31-frameTolerance:
            # keep track of start time/frame for later
            encoding_stop.frameNStart = frameN  # exact frame index
```

```
    encoding_stop.tStart = t  # local t and not account for scr refresh
    encoding_stop.tStartRefresh = tThisFlipGlobal  # on global time
    encoding_stop.play(when=win)  # sync with win flip
if encoding_stop.status == STARTED:
    # is it time to stop? (based on global clock, using actual start)
    if tThisFlipGlobal > encoding_stop.tStartRefresh + 1-frameTolerance:
        # keep track of stop time/frame for later
        encoding_stop.tStop = t  # not accounting for scr refresh
        encoding_stop.frameNStop = frameN  # exact frame index
        win.timeOnFlip(encoding_stop, 'tStopRefresh')  # time at next scr refresh
        encoding_stop.stop()

# *button1_1* updates
if button1_1.status == NOT_STARTED and tThisFlip >= 1-frameTolerance:
    # keep track of start time/frame for later
    button1_1.frameNStart = frameN  # exact frame index
    button1_1.tStart = t  # local t and not account for scr refresh
    button1_1.tStartRefresh = tThisFlipGlobal  # on global time
    win.timeOnFlip(button1_1, 'tStartRefresh')  # time at next scr refresh
    button1_1.setAutoDraw(True)
if button1_1.status == STARTED:
    # is it time to stop? (based on global clock, using actual start)
    if tThisFlipGlobal > button1_1.tStartRefresh + 30-frameTolerance:
        # keep track of stop time/frame for later
        button1_1.tStop = t  # not accounting for scr refresh
        button1_1.frameNStop = frameN  # exact frame index
        win.timeOnFlip(button1_1, 'tStopRefresh')  # time at next scr refresh
        button1_1.setAutoDraw(False)
if button1_1.status == STARTED:
    # check whether button1_1 has been pressed
    if button1_1.isClicked:
        if not button1_1.wasClicked:
            button1_1.timesOn.append(button1_1.buttonClock.getTime()) # store time of first click
            button1_1.timesOff.append(button1_1.buttonClock.getTime()) # store time clicked until
        else:
            button1_1.timesOff[-1] = button1_1.buttonClock.getTime() # update time clicked until
        None
        button1_1.wasClicked = True  # if button1_1 is still clicked next frame, it is not a new click
    else:
        button1_1.wasClicked = False  # if button1_1 is clicked next frame, it is a new click
else:
    button1_1.wasClicked = False  # if button1_1 is clicked next frame, it is a new click

# *button1_2* updates
if button1_2.status == NOT_STARTED and tThisFlip >= 1-frameTolerance:
    # keep track of start time/frame for later
    button1_2.frameNStart = frameN  # exact frame index
    button1_2.tStart = t  # local t and not account for scr refresh
    button1_2.tStartRefresh = tThisFlipGlobal  # on global time
    win.timeOnFlip(button1_2, 'tStartRefresh')  # time at next scr refresh
    button1_2.setAutoDraw(True)
if button1_2.status == STARTED:
    # is it time to stop? (based on global clock, using actual start)
    if tThisFlipGlobal > button1_2.tStartRefresh + 30-frameTolerance:
        # keep track of stop time/frame for later
        button1_2.tStop = t  # not accounting for scr refresh
        button1_2.frameNStop = frameN  # exact frame index
        win.timeOnFlip(button1_2, 'tStopRefresh')  # time at next scr refresh
        button1_2.setAutoDraw(False)
```

```
if button1_2.status == STARTED:
    # check whether button1_2 has been pressed
    if button1_2.isClicked:
        if not button1_2.wasClicked:
            button1_2.timesOn.append(button1_2.buttonClock.getTime()) # store time of first click
            button1_2.timesOff.append(button1_2.buttonClock.getTime()) # store time clicked until
        else:
            button1_2.timesOff[-1] = button1_2.buttonClock.getTime() # update time clicked until
        None
        button1_2.wasClicked = True  # if button1_2 is still clicked next frame, it is not a new click
    else:
        button1_2.wasClicked = False  # if button1_2 is clicked next frame, it is a new click
else:
    button1_2.wasClicked = False  # if button1_2 is clicked next frame, it is a new click

# *button1_3* updates
if button1_3.status == NOT_STARTED and tThisFlip >= 1-frameTolerance:
    # keep track of start time/frame for later
    button1_3.frameNStart = frameN  # exact frame index
    button1_3.tStart = t  # local t and not account for scr refresh
    button1_3.tStartRefresh = tThisFlipGlobal  # on global time
    win.timeOnFlip(button1_3, 'tStartRefresh')  # time at next scr refresh
    button1_3.setAutoDraw(True)
if button1_3.status == STARTED:
    # is it time to stop? (based on global clock, using actual start)
    if tThisFlipGlobal > button1_3.tStartRefresh + 30-frameTolerance:
        # keep track of stop time/frame for later
        button1_3.tStop = t  # not accounting for scr refresh
        button1_3.frameNStop = frameN  # exact frame index
        win.timeOnFlip(button1_3, 'tStopRefresh')  # time at next scr refresh
        button1_3.setAutoDraw(False)
if button1_3.status == STARTED:
    # check whether button1_3 has been pressed
    if button1_3.isClicked:
        if not button1_3.wasClicked:
            button1_3.timesOn.append(button1_3.buttonClock.getTime()) # store time of first click
            button1_3.timesOff.append(button1_3.buttonClock.getTime()) # store time clicked until
        else:
            button1_3.timesOff[-1] = button1_3.buttonClock.getTime() # update time clicked until
        None
        button1_3.wasClicked = True  # if button1_3 is still clicked next frame, it is not a new click
    else:
        button1_3.wasClicked = False  # if button1_3 is clicked next frame, it is a new click
else:
    button1_3.wasClicked = False  # if button1_3 is clicked next frame, it is a new click

# *button1_4* updates
if button1_4.status == NOT_STARTED and tThisFlip >= 1-frameTolerance:
    # keep track of start time/frame for later
    button1_4.frameNStart = frameN  # exact frame index
    button1_4.tStart = t  # local t and not account for scr refresh
    button1_4.tStartRefresh = tThisFlipGlobal  # on global time
    win.timeOnFlip(button1_4, 'tStartRefresh')  # time at next scr refresh
    button1_4.setAutoDraw(True)
if button1_4.status == STARTED:
    # is it time to stop? (based on global clock, using actual start)
    if tThisFlipGlobal > button1_4.tStartRefresh + 30-frameTolerance:
        # keep track of stop time/frame for later
        button1_4.tStop = t  # not accounting for scr refresh
```

```
        button1_4.frameNStop = frameN  # exact frame index
        win.timeOnFlip(button1_4, 'tStopRefresh')  # time at next scr refresh
        button1_4.setAutoDraw(False)
    if button1_4.status == STARTED:
        # check whether button1_4 has been pressed
        if button1_4.isClicked:
            if not button1_4.wasClicked:
                button1_4.timesOn.append(button1_4.buttonClock.getTime()) # store time of first click
                button1_4.timesOff.append(button1_4.buttonClock.getTime()) # store time clicked until
            else:
                button1_4.timesOff[-1] = button1_4.buttonClock.getTime() # update time clicked until
            None
            button1_4.wasClicked = True  # if button1_4 is still clicked next frame, it is not a new click
        else:
            button1_4.wasClicked = False  # if button1_4 is clicked next frame, it is a new click
    else:
        button1_4.wasClicked = False  # if button1_4 is clicked next frame, it is a new click

    # *button1_5* updates
    if button1_5.status == NOT_STARTED and tThisFlip >= 1-frameTolerance:
        # keep track of start time/frame for later
        button1_5.frameNStart = frameN  # exact frame index
        button1_5.tStart = t  # local t and not account for scr refresh
        button1_5.tStartRefresh = tThisFlipGlobal  # on global time
        win.timeOnFlip(button1_5, 'tStartRefresh')  # time at next scr refresh
        button1_5.setAutoDraw(True)
    if button1_5.status == STARTED:
        # is it time to stop? (based on global clock, using actual start)
        if tThisFlipGlobal > button1_5.tStartRefresh + 30-frameTolerance:
            # keep track of stop time/frame for later
            button1_5.tStop = t  # not accounting for scr refresh
            button1_5.frameNStop = frameN  # exact frame index
            win.timeOnFlip(button1_5, 'tStopRefresh')  # time at next scr refresh
            button1_5.setAutoDraw(False)
    if button1_5.status == STARTED:
        # check whether button1_5 has been pressed
        if button1_5.isClicked:
            if not button1_5.wasClicked:
                button1_5.timesOn.append(button1_5.buttonClock.getTime()) # store time of first click
                button1_5.timesOff.append(button1_5.buttonClock.getTime()) # store time clicked until
            else:
                button1_5.timesOff[-1] = button1_5.buttonClock.getTime() # update time clicked until
            None
            button1_5.wasClicked = True  # if button1_5 is still clicked next frame, it is not a new click
        else:
            button1_5.wasClicked = False  # if button1_5 is clicked next frame, it is a new click
    else:
        button1_5.wasClicked = False  # if button1_5 is clicked next frame, it is a new click

    # *button1_6* updates
    if button1_6.status == NOT_STARTED and tThisFlip >= 1-frameTolerance:
        # keep track of start time/frame for later
        button1_6.frameNStart = frameN  # exact frame index
        button1_6.tStart = t  # local t and not account for scr refresh
        button1_6.tStartRefresh = tThisFlipGlobal  # on global time
        win.timeOnFlip(button1_6, 'tStartRefresh')  # time at next scr refresh
        button1_6.setAutoDraw(True)
    if button1_6.status == STARTED:
        # is it time to stop? (based on global clock, using actual start)
```

```
    if tThisFlipGlobal > button1_6.tStartRefresh + 30-frameTolerance:
        # keep track of stop time/frame for later
        button1_6.tStop = t  # not accounting for scr refresh
        button1_6.frameNStop = frameN  # exact frame index
        win.timeOnFlip(button1_6, 'tStopRefresh')  # time at next scr refresh
        button1_6.setAutoDraw(False)
if button1_6.status == STARTED:
    # check whether button1_6 has been pressed
    if button1_6.isClicked:
        if not button1_6.wasClicked:
            button1_6.timesOn.append(button1_6.buttonClock.getTime()) # store time of first click
            button1_6.timesOff.append(button1_6.buttonClock.getTime()) # store time clicked until
        else:
            button1_6.timesOff[-1] = button1_6.buttonClock.getTime() # update time clicked until
        None
        button1_6.wasClicked = True  # if button1_6 is still clicked next frame, it is not a new click
    else:
        button1_6.wasClicked = False  # if button1_6 is clicked next frame, it is a new click
else:
    button1_6.wasClicked = False  # if button1_6 is clicked next frame, it is a new click

# *button2_1* updates
if button2_1.status == NOT_STARTED and tThisFlip >= 1-frameTolerance:
    # keep track of start time/frame for later
    button2_1.frameNStart = frameN  # exact frame index
    button2_1.tStart = t  # local t and not account for scr refresh
    button2_1.tStartRefresh = tThisFlipGlobal  # on global time
    win.timeOnFlip(button2_1, 'tStartRefresh')  # time at next scr refresh
    button2_1.setAutoDraw(True)
if button2_1.status == STARTED:
    # is it time to stop? (based on global clock, using actual start)
    if tThisFlipGlobal > button2_1.tStartRefresh + 30-frameTolerance:
        # keep track of stop time/frame for later
        button2_1.tStop = t  # not accounting for scr refresh
        button2_1.frameNStop = frameN  # exact frame index
        win.timeOnFlip(button2_1, 'tStopRefresh')  # time at next scr refresh
        button2_1.setAutoDraw(False)
if button2_1.status == STARTED:
    # check whether button2_1 has been pressed
    if button2_1.isClicked:
        if not button2_1.wasClicked:
            button2_1.timesOn.append(button2_1.buttonClock.getTime()) # store time of first click
            button2_1.timesOff.append(button2_1.buttonClock.getTime()) # store time clicked until
        else:
            button2_1.timesOff[-1] = button2_1.buttonClock.getTime() # update time clicked until
        None
        button2_1.wasClicked = True  # if button2_1 is still clicked next frame, it is not a new click
    else:
        button2_1.wasClicked = False  # if button2_1 is clicked next frame, it is a new click
else:
    button2_1.wasClicked = False  # if button2_1 is clicked next frame, it is a new click

# *button2_2* updates
if button2_2.status == NOT_STARTED and tThisFlip >= 1-frameTolerance:
    # keep track of start time/frame for later
    button2_2.frameNStart = frameN  # exact frame index
    button2_2.tStart = t  # local t and not account for scr refresh
    button2_2.tStartRefresh = tThisFlipGlobal  # on global time
    win.timeOnFlip(button2_2, 'tStartRefresh')  # time at next scr refresh
```

```
    button2_2.setAutoDraw(True)
if button2_2.status == STARTED:
    # is it time to stop? (based on global clock, using actual start)
    if tThisFlipGlobal > button2_2.tStartRefresh + 30-frameTolerance:
        # keep track of stop time/frame for later
        button2_2.tStop = t  # not accounting for scr refresh
        button2_2.frameNStop = frameN  # exact frame index
        win.timeOnFlip(button2_2, 'tStopRefresh')  # time at next scr refresh
        button2_2.setAutoDraw(False)
if button2_2.status == STARTED:
    # check whether button2_2 has been pressed
    if button2_2.isClicked:
        if not button2_2.wasClicked:
            button2_2.timesOn.append(button2_2.buttonClock.getTime()) # store time of first click
            button2_2.timesOff.append(button2_2.buttonClock.getTime()) # store time clicked until
        else:
            button2_2.timesOff[-1] = button2_2.buttonClock.getTime() # update time clicked until
        None
        button2_2.wasClicked = True  # if button2_2 is still clicked next frame, it is not a new click
    else:
        button2_2.wasClicked = False  # if button2_2 is clicked next frame, it is a new click
else:
    button2_2.wasClicked = False  # if button2_2 is clicked next frame, it is a new click

# *button2_3* updates
if button2_3.status == NOT_STARTED and tThisFlip >= 1-frameTolerance:
    # keep track of start time/frame for later
    button2_3.frameNStart = frameN  # exact frame index
    button2_3.tStart = t  # local t and not account for scr refresh
    button2_3.tStartRefresh = tThisFlipGlobal  # on global time
    win.timeOnFlip(button2_3, 'tStartRefresh')  # time at next scr refresh
    button2_3.setAutoDraw(True)
if button2_3.status == STARTED:
    # is it time to stop? (based on global clock, using actual start)
    if tThisFlipGlobal > button2_3.tStartRefresh + 30-frameTolerance:
        # keep track of stop time/frame for later
        button2_3.tStop = t  # not accounting for scr refresh
        button2_3.frameNStop = frameN  # exact frame index
        win.timeOnFlip(button2_3, 'tStopRefresh')  # time at next scr refresh
        button2_3.setAutoDraw(False)
if button2_3.status == STARTED:
    # check whether button2_3 has been pressed
    if button2_3.isClicked:
        if not button2_3.wasClicked:
            button2_3.timesOn.append(button2_3.buttonClock.getTime()) # store time of first click
            button2_3.timesOff.append(button2_3.buttonClock.getTime()) # store time clicked until
        else:
            button2_3.timesOff[-1] = button2_3.buttonClock.getTime() # update time clicked until
        None
        button2_3.wasClicked = True  # if button2_3 is still clicked next frame, it is not a new click
    else:
        button2_3.wasClicked = False  # if button2_3 is clicked next frame, it is a new click
else:
    button2_3.wasClicked = False  # if button2_3 is clicked next frame, it is a new click

# *button2_4* updates
if button2_4.status == NOT_STARTED and tThisFlip >= 1-frameTolerance:
    # keep track of start time/frame for later
    button2_4.frameNStart = frameN  # exact frame index
```

```
    button2_4.tStart = t  # local t and not account for scr refresh
    button2_4.tStartRefresh = tThisFlipGlobal  # on global time
    win.timeOnFlip(button2_4, 'tStartRefresh')  # time at next scr refresh
    button2_4.setAutoDraw(True)
  if button2_4.status == STARTED:
    # is it time to stop? (based on global clock, using actual start)
    if tThisFlipGlobal > button2_4.tStartRefresh + 30-frameTolerance:
      # keep track of stop time/frame for later
      button2_4.tStop = t  # not accounting for scr refresh
      button2_4.frameNStop = frameN  # exact frame index
      win.timeOnFlip(button2_4, 'tStopRefresh')  # time at next scr refresh
      button2_4.setAutoDraw(False)
  if button2_4.status == STARTED:
    # check whether button2_4 has been pressed
    if button2_4.isClicked:
      if not button2_4.wasClicked:
        button2_4.timesOn.append(button2_4.buttonClock.getTime()) # store time of first click
        button2_4.timesOff.append(button2_4.buttonClock.getTime()) # store time clicked until
      else:
        button2_4.timesOff[-1] = button2_4.buttonClock.getTime() # update time clicked until
      None
      button2_4.wasClicked = True  # if button2_4 is still clicked next frame, it is not a new click
    else:
      button2_4.wasClicked = False  # if button2_4 is clicked next frame, it is a new click
  else:
    button2_4.wasClicked = False  # if button2_4 is clicked next frame, it is a new click

  # *button2_5* updates
  if button2_5.status == NOT_STARTED and tThisFlip >= 1-frameTolerance:
    # keep track of start time/frame for later
    button2_5.frameNStart = frameN  # exact frame index
    button2_5.tStart = t  # local t and not account for scr refresh
    button2_5.tStartRefresh = tThisFlipGlobal  # on global time
    win.timeOnFlip(button2_5, 'tStartRefresh')  # time at next scr refresh
    button2_5.setAutoDraw(True)
  if button2_5.status == STARTED:
    # is it time to stop? (based on global clock, using actual start)
    if tThisFlipGlobal > button2_5.tStartRefresh + 30-frameTolerance:
      # keep track of stop time/frame for later
      button2_5.tStop = t  # not accounting for scr refresh
      button2_5.frameNStop = frameN  # exact frame index
      win.timeOnFlip(button2_5, 'tStopRefresh')  # time at next scr refresh
      button2_5.setAutoDraw(False)
  if button2_5.status == STARTED:
    # check whether button2_5 has been pressed
    if button2_5.isClicked:
      if not button2_5.wasClicked:
        button2_5.timesOn.append(button2_5.buttonClock.getTime()) # store time of first click
        button2_5.timesOff.append(button2_5.buttonClock.getTime()) # store time clicked until
      else:
        button2_5.timesOff[-1] = button2_5.buttonClock.getTime() # update time clicked until
      None
      button2_5.wasClicked = True  # if button2_5 is still clicked next frame, it is not a new click
    else:
      button2_5.wasClicked = False  # if button2_5 is clicked next frame, it is a new click
  else:
    button2_5.wasClicked = False  # if button2_5 is clicked next frame, it is a new click

  # *button2_6* updates
```

```
    if button2_6.status == NOT_STARTED and tThisFlip >= 1-frameTolerance:
        # keep track of start time/frame for later
        button2_6.frameNStart = frameN  # exact frame index
        button2_6.tStart = t  # local t and not account for scr refresh
        button2_6.tStartRefresh = tThisFlipGlobal  # on global time
        win.timeOnFlip(button2_6, 'tStartRefresh')  # time at next scr refresh
        button2_6.setAutoDraw(True)
    if button2_6.status == STARTED:
        # is it time to stop? (based on global clock, using actual start)
        if tThisFlipGlobal > button2_6.tStartRefresh + 30-frameTolerance:
            # keep track of stop time/frame for later
            button2_6.tStop = t  # not accounting for scr refresh
            button2_6.frameNStop = frameN  # exact frame index
            win.timeOnFlip(button2_6, 'tStopRefresh')  # time at next scr refresh
            button2_6.setAutoDraw(False)
    if button2_6.status == STARTED:
        # check whether button2_6 has been pressed
        if button2_6.isClicked:
            if not button2_6.wasClicked:
                button2_6.timesOn.append(button2_6.buttonClock.getTime()) # store time of first click
                button2_6.timesOff.append(button2_6.buttonClock.getTime()) # store time clicked until
            else:
                button2_6.timesOff[-1] = button2_6.buttonClock.getTime() # update time clicked until
            None
            button2_6.wasClicked = True  # if button2_6 is still clicked next frame, it is not a new click
        else:
            button2_6.wasClicked = False  # if button2_6 is clicked next frame, it is a new click
    else:
        button2_6.wasClicked = False  # if button2_6 is clicked next frame, it is a new click

    # *button3_1* updates
    if button3_1.status == NOT_STARTED and tThisFlip >= 1-frameTolerance:
        # keep track of start time/frame for later
        button3_1.frameNStart = frameN  # exact frame index
        button3_1.tStart = t  # local t and not account for scr refresh
        button3_1.tStartRefresh = tThisFlipGlobal  # on global time
        win.timeOnFlip(button3_1, 'tStartRefresh')  # time at next scr refresh
        button3_1.setAutoDraw(True)
    if button3_1.status == STARTED:
        # is it time to stop? (based on global clock, using actual start)
        if tThisFlipGlobal > button3_1.tStartRefresh + 30-frameTolerance:
            # keep track of stop time/frame for later
            button3_1.tStop = t  # not accounting for scr refresh
            button3_1.frameNStop = frameN  # exact frame index
            win.timeOnFlip(button3_1, 'tStopRefresh')  # time at next scr refresh
            button3_1.setAutoDraw(False)
    if button3_1.status == STARTED:
        # check whether button3_1 has been pressed
        if button3_1.isClicked:
            if not button3_1.wasClicked:
                button3_1.timesOn.append(button3_1.buttonClock.getTime()) # store time of first click
                button3_1.timesOff.append(button3_1.buttonClock.getTime()) # store time clicked until
            else:
                button3_1.timesOff[-1] = button3_1.buttonClock.getTime() # update time clicked until
            None
            button3_1.wasClicked = True  # if button3_1 is still clicked next frame, it is not a new click
        else:
            button3_1.wasClicked = False  # if button3_1 is clicked next frame, it is a new click
    else:
```

```
    button3_1.wasClicked = False  # if button3_1 is clicked next frame, it is a new click

# *button3_2* updates
if button3_2.status == NOT_STARTED and tThisFlip >= 1-frameTolerance:
    # keep track of start time/frame for later
    button3_2.frameNStart = frameN  # exact frame index
    button3_2.tStart = t  # local t and not account for scr refresh
    button3_2.tStartRefresh = tThisFlipGlobal  # on global time
    win.timeOnFlip(button3_2, 'tStartRefresh')  # time at next scr refresh
    button3_2.setAutoDraw(True)
if button3_2.status == STARTED:
    # is it time to stop? (based on global clock, using actual start)
    if tThisFlipGlobal > button3_2.tStartRefresh + 30-frameTolerance:
        # keep track of stop time/frame for later
        button3_2.tStop = t  # not accounting for scr refresh
        button3_2.frameNStop = frameN  # exact frame index
        win.timeOnFlip(button3_2, 'tStopRefresh')  # time at next scr refresh
        button3_2.setAutoDraw(False)
if button3_2.status == STARTED:
    # check whether button3_2 has been pressed
    if button3_2.isClicked:
        if not button3_2.wasClicked:
            button3_2.timesOn.append(button3_2.buttonClock.getTime()) # store time of first click
            button3_2.timesOff.append(button3_2.buttonClock.getTime()) # store time clicked until
        else:
            button3_2.timesOff[-1] = button3_2.buttonClock.getTime() # update time clicked until
        None
        button3_2.wasClicked = True  # if button3_2 is still clicked next frame, it is not a new click
    else:
        button3_2.wasClicked = False  # if button3_2 is clicked next frame, it is a new click
else:
    button3_2.wasClicked = False  # if button3_2 is clicked next frame, it is a new click

# *button3_3* updates
if button3_3.status == NOT_STARTED and tThisFlip >= 1-frameTolerance:
    # keep track of start time/frame for later
    button3_3.frameNStart = frameN  # exact frame index
    button3_3.tStart = t  # local t and not account for scr refresh
    button3_3.tStartRefresh = tThisFlipGlobal  # on global time
    win.timeOnFlip(button3_3, 'tStartRefresh')  # time at next scr refresh
    button3_3.setAutoDraw(True)
if button3_3.status == STARTED:
    # is it time to stop? (based on global clock, using actual start)
    if tThisFlipGlobal > button3_3.tStartRefresh + 30-frameTolerance:
        # keep track of stop time/frame for later
        button3_3.tStop = t  # not accounting for scr refresh
        button3_3.frameNStop = frameN  # exact frame index
        win.timeOnFlip(button3_3, 'tStopRefresh')  # time at next scr refresh
        button3_3.setAutoDraw(False)
if button3_3.status == STARTED:
    # check whether button3_3 has been pressed
    if button3_3.isClicked:
        if not button3_3.wasClicked:
            button3_3.timesOn.append(button3_3.buttonClock.getTime()) # store time of first click
            button3_3.timesOff.append(button3_3.buttonClock.getTime()) # store time clicked until
        else:
            button3_3.timesOff[-1] = button3_3.buttonClock.getTime() # update time clicked until
        None
        button3_3.wasClicked = True  # if button3_3 is still clicked next frame, it is not a new click
```

```
    else:
        button3_3.wasClicked = False  # if button3_3 is clicked next frame, it is a new click
else:
    button3_3.wasClicked = False  # if button3_3 is clicked next frame, it is a new click
# *button3_4* updates
if button3_4.status == NOT_STARTED and tThisFlip >= 1-frameTolerance:
    # keep track of start time/frame for later
    button3_4.frameNStart = frameN  # exact frame index
    button3_4.tStart = t  # local t and not account for scr refresh
    button3_4.tStartRefresh = tThisFlipGlobal  # on global time
    win.timeOnFlip(button3_4, 'tStartRefresh')  # time at next scr refresh
    button3_4.setAutoDraw(True)
if button3_4.status == STARTED:
    # is it time to stop? (based on global clock, using actual start)
    if tThisFlipGlobal > button3_4.tStartRefresh + 30-frameTolerance:
        # keep track of stop time/frame for later
        button3_4.tStop = t  # not accounting for scr refresh
        button3_4.frameNStop = frameN  # exact frame index
        win.timeOnFlip(button3_4, 'tStopRefresh')  # time at next scr refresh
        button3_4.setAutoDraw(False)
if button3_4.status == STARTED:
    # check whether button3_4 has been pressed
    if button3_4.isClicked:
        if not button3_4.wasClicked:
            button3_4.timesOn.append(button3_4.buttonClock.getTime()) # store time of first click
            button3_4.timesOff.append(button3_4.buttonClock.getTime()) # store time clicked until
        else:
            button3_4.timesOff[-1] = button3_4.buttonClock.getTime() # update time clicked until
        None
        button3_4.wasClicked = True  # if button3_4 is still clicked next frame, it is not a new click
    else:
        button3_4.wasClicked = False  # if button3_4 is clicked next frame, it is a new click
else:
    button3_4.wasClicked = False  # if button3_4 is clicked next frame, it is a new click

# *button3_5* updates
if button3_5.status == NOT_STARTED and tThisFlip >= 1-frameTolerance:
    # keep track of start time/frame for later
    button3_5.frameNStart = frameN  # exact frame index
    button3_5.tStart = t  # local t and not account for scr refresh
    button3_5.tStartRefresh = tThisFlipGlobal  # on global time
    win.timeOnFlip(button3_5, 'tStartRefresh')  # time at next scr refresh
    button3_5.setAutoDraw(True)
if button3_5.status == STARTED:
    # is it time to stop? (based on global clock, using actual start)
    if tThisFlipGlobal > button3_5.tStartRefresh + 30-frameTolerance:
        # keep track of stop time/frame for later
        button3_5.tStop = t  # not accounting for scr refresh
        button3_5.frameNStop = frameN  # exact frame index
        win.timeOnFlip(button3_5, 'tStopRefresh')  # time at next scr refresh
        button3_5.setAutoDraw(False)
if button3_5.status == STARTED:
    # check whether button3_5 has been pressed
    if button3_5.isClicked:
        if not button3_5.wasClicked:
            button3_5.timesOn.append(button3_5.buttonClock.getTime()) # store time of first click
            button3_5.timesOff.append(button3_5.buttonClock.getTime()) # store time clicked until
        else:
            button3_5.timesOff[-1] = button3_5.buttonClock.getTime() # update time clicked until
```

```
        None
        button3_5.wasClicked = True  # if button3_5 is still clicked next frame, it is not a new click
    else:
        button3_5.wasClicked = False  # if button3_5 is clicked next frame, it is a new click
else:
    button3_5.wasClicked = False  # if button3_5 is clicked next frame, it is a new click

# *button3_6* updates
if button3_6.status == NOT_STARTED and tThisFlip >= 1-frameTolerance:
    # keep track of start time/frame for later
    button3_6.frameNStart = frameN  # exact frame index
    button3_6.tStart = t  # local t and not account for scr refresh
    button3_6.tStartRefresh = tThisFlipGlobal  # on global time
    win.timeOnFlip(button3_6, 'tStartRefresh')  # time at next scr refresh
    button3_6.setAutoDraw(True)
if button3_6.status == STARTED:
    # is it time to stop? (based on global clock, using actual start)
    if tThisFlipGlobal > button3_6.tStartRefresh + 30-frameTolerance:
        # keep track of stop time/frame for later
        button3_6.tStop = t  # not accounting for scr refresh
        button3_6.frameNStop = frameN  # exact frame index
        win.timeOnFlip(button3_6, 'tStopRefresh')  # time at next scr refresh
        button3_6.setAutoDraw(False)
if button3_6.status == STARTED:
    # check whether button3_6 has been pressed
    if button3_6.isClicked:
        if not button3_6.wasClicked:
            button3_6.timesOn.append(button3_6.buttonClock.getTime()) # store time of first click
            button3_6.timesOff.append(button3_6.buttonClock.getTime()) # store time clicked until
        else:
            button3_6.timesOff[-1] = button3_6.buttonClock.getTime() # update time clicked until
        None
        button3_6.wasClicked = True  # if button3_6 is still clicked next frame, it is not a new click
    else:
        button3_6.wasClicked = False  # if button3_6 is clicked next frame, it is a new click
else:
    button3_6.wasClicked = False  # if button3_6 is clicked next frame, it is a new click

# *button4_1* updates
if button4_1.status == NOT_STARTED and tThisFlip >= 1-frameTolerance:
    # keep track of start time/frame for later
    button4_1.frameNStart = frameN  # exact frame index
    button4_1.tStart = t  # local t and not account for scr refresh
    button4_1.tStartRefresh = tThisFlipGlobal  # on global time
    win.timeOnFlip(button4_1, 'tStartRefresh')  # time at next scr refresh
    button4_1.setAutoDraw(True)
if button4_1.status == STARTED:
    # is it time to stop? (based on global clock, using actual start)
    if tThisFlipGlobal > button4_1.tStartRefresh + 30-frameTolerance:
        # keep track of stop time/frame for later
        button4_1.tStop = t  # not accounting for scr refresh
        button4_1.frameNStop = frameN  # exact frame index
        win.timeOnFlip(button4_1, 'tStopRefresh')  # time at next scr refresh
        button4_1.setAutoDraw(False)
if button4_1.status == STARTED:
    # check whether button4_1 has been pressed
    if button4_1.isClicked:
        if not button4_1.wasClicked:
            button4_1.timesOn.append(button4_1.buttonClock.getTime()) # store time of first click
```

```
            button4_1.timesOff.append(button4_1.buttonClock.getTime()) # store time clicked until
        else:
            button4_1.timesOff[-1] = button4_1.buttonClock.getTime() # update time clicked until
        None
        button4_1.wasClicked = True  # if button4_1 is still clicked next frame, it is not a new click
    else:
        button4_1.wasClicked = False  # if button4_1 is clicked next frame, it is a new click
else:
    button4_1.wasClicked = False  # if button4_1 is clicked next frame, it is a new click

# *button4_2* updates
if button4_2.status == NOT_STARTED and tThisFlip >= 1-frameTolerance:
    # keep track of start time/frame for later
    button4_2.frameNStart = frameN  # exact frame index
    button4_2.tStart = t  # local t and not account for scr refresh
    button4_2.tStartRefresh = tThisFlipGlobal  # on global time
    win.timeOnFlip(button4_2, 'tStartRefresh')  # time at next scr refresh
    button4_2.setAutoDraw(True)
if button4_2.status == STARTED:
    # is it time to stop? (based on global clock, using actual start)
    if tThisFlipGlobal > button4_2.tStartRefresh + 30-frameTolerance:
        # keep track of stop time/frame for later
        button4_2.tStop = t  # not accounting for scr refresh
        button4_2.frameNStop = frameN  # exact frame index
        win.timeOnFlip(button4_2, 'tStopRefresh')  # time at next scr refresh
        button4_2.setAutoDraw(False)
if button4_2.status == STARTED:
    # check whether button4_2 has been pressed
    if button4_2.isClicked:
        if not button4_2.wasClicked:
            button4_2.timesOn.append(button4_2.buttonClock.getTime()) # store time of first click
            button4_2.timesOff.append(button4_2.buttonClock.getTime()) # store time clicked until
        else:
            button4_2.timesOff[-1] = button4_2.buttonClock.getTime() # update time clicked until
        None
        button4_2.wasClicked = True  # if button4_2 is still clicked next frame, it is not a new click
    else:
        button4_2.wasClicked = False  # if button4_2 is clicked next frame, it is a new click
else:
    button4_2.wasClicked = False  # if button4_2 is clicked next frame, it is a new click

# *button4_3* updates
if button4_3.status == NOT_STARTED and tThisFlip >= 1-frameTolerance:
    # keep track of start time/frame for later
    button4_3.frameNStart = frameN  # exact frame index
    button4_3.tStart = t  # local t and not account for scr refresh
    button4_3.tStartRefresh = tThisFlipGlobal  # on global time
    win.timeOnFlip(button4_3, 'tStartRefresh')  # time at next scr refresh
    button4_3.setAutoDraw(True)
if button4_3.status == STARTED:
    # is it time to stop? (based on global clock, using actual start)
    if tThisFlipGlobal > button4_3.tStartRefresh + 30-frameTolerance:
        # keep track of stop time/frame for later
        button4_3.tStop = t  # not accounting for scr refresh
        button4_3.frameNStop = frameN  # exact frame index
        win.timeOnFlip(button4_3, 'tStopRefresh')  # time at next scr refresh
        button4_3.setAutoDraw(False)
if button4_3.status == STARTED:
    # check whether button4_3 has been pressed
```

```
    if button4_3.isClicked:
        if not button4_3.wasClicked:
            button4_3.timesOn.append(button4_3.buttonClock.getTime()) # store time of first click
            button4_3.timesOff.append(button4_3.buttonClock.getTime()) # store time clicked until
        else:
            button4_3.timesOff[-1] = button4_3.buttonClock.getTime() # update time clicked until
        None
        button4_3.wasClicked = True  # if button4_3 is still clicked next frame, it is not a new click
    else:
        button4_3.wasClicked = False  # if button4_3 is clicked next frame, it is a new click
else:
    button4_3.wasClicked = False  # if button4_3 is clicked next frame, it is a new click

# *button4_4* updates
if button4_4.status == NOT_STARTED and tThisFlip >= 1-frameTolerance:
    # keep track of start time/frame for later
    button4_4.frameNStart = frameN  # exact frame index
    button4_4.tStart = t  # local t and not account for scr refresh
    button4_4.tStartRefresh = tThisFlipGlobal  # on global time
    win.timeOnFlip(button4_4, 'tStartRefresh')  # time at next scr refresh
    button4_4.setAutoDraw(True)
if button4_4.status == STARTED:
    # is it time to stop? (based on global clock, using actual start)
    if tThisFlipGlobal > button4_4.tStartRefresh + 30-frameTolerance:
        # keep track of stop time/frame for later
        button4_4.tStop = t  # not accounting for scr refresh
        button4_4.frameNStop = frameN  # exact frame index
        win.timeOnFlip(button4_4, 'tStopRefresh')  # time at next scr refresh
        button4_4.setAutoDraw(False)
if button4_4.status == STARTED:
    # check whether button4_4 has been pressed
    if button4_4.isClicked:
        if not button4_4.wasClicked:
            button4_4.timesOn.append(button4_4.buttonClock.getTime()) # store time of first click
            button4_4.timesOff.append(button4_4.buttonClock.getTime()) # store time clicked until
        else:
            button4_4.timesOff[-1] = button4_4.buttonClock.getTime() # update time clicked until
        None
        button4_4.wasClicked = True  # if button4_4 is still clicked next frame, it is not a new click
    else:
        button4_4.wasClicked = False  # if button4_4 is clicked next frame, it is a new click
else:
    button4_4.wasClicked = False  # if button4_4 is clicked next frame, it is a new click

# *button4_5* updates
if button4_5.status == NOT_STARTED and tThisFlip >= 1-frameTolerance:
    # keep track of start time/frame for later
    button4_5.frameNStart = frameN  # exact frame index
    button4_5.tStart = t  # local t and not account for scr refresh
    button4_5.tStartRefresh = tThisFlipGlobal  # on global time
    win.timeOnFlip(button4_5, 'tStartRefresh')  # time at next scr refresh
    button4_5.setAutoDraw(True)
if button4_5.status == STARTED:
    # is it time to stop? (based on global clock, using actual start)
    if tThisFlipGlobal > button4_5.tStartRefresh + 30-frameTolerance:
        # keep track of stop time/frame for later
        button4_5.tStop = t  # not accounting for scr refresh
        button4_5.frameNStop = frameN  # exact frame index
        win.timeOnFlip(button4_5, 'tStopRefresh')  # time at next scr refresh
```

```
        button4_5.setAutoDraw(False)
    if button4_5.status == STARTED:
        # check whether button4_5 has been pressed
        if button4_5.isClicked:
            if not button4_5.wasClicked:
                button4_5.timesOn.append(button4_5.buttonClock.getTime()) # store time of first click
                button4_5.timesOff.append(button4_5.buttonClock.getTime()) # store time clicked until
            else:
                button4_5.timesOff[-1] = button4_5.buttonClock.getTime() # update time clicked until
            None
            button4_5.wasClicked = True  # if button4_5 is still clicked next frame, it is not a new click
        else:
            button4_5.wasClicked = False  # if button4_5 is clicked next frame, it is a new click
    else:
        button4_5.wasClicked = False  # if button4_5 is clicked next frame, it is a new click

    # *button4_6* updates
    if button4_6.status == NOT_STARTED and tThisFlip >= 1-frameTolerance:
        # keep track of start time/frame for later
        button4_6.frameNStart = frameN  # exact frame index
        button4_6.tStart = t  # local t and not account for scr refresh
        button4_6.tStartRefresh = tThisFlipGlobal  # on global time
        win.timeOnFlip(button4_6, 'tStartRefresh')  # time at next scr refresh
        button4_6.setAutoDraw(True)
    if button4_6.status == STARTED:
        # is it time to stop? (based on global clock, using actual start)
        if tThisFlipGlobal > button4_6.tStartRefresh + 30-frameTolerance:
            # keep track of stop time/frame for later
            button4_6.tStop = t  # not accounting for scr refresh
            button4_6.frameNStop = frameN  # exact frame index
            win.timeOnFlip(button4_6, 'tStopRefresh')  # time at next scr refresh
            button4_6.setAutoDraw(False)
    if button4_6.status == STARTED:
        # check whether button4_6 has been pressed
        if button4_6.isClicked:
            if not button4_6.wasClicked:
                button4_6.timesOn.append(button4_6.buttonClock.getTime()) # store time of first click
                button4_6.timesOff.append(button4_6.buttonClock.getTime()) # store time clicked until
            else:
                button4_6.timesOff[-1] = button4_6.buttonClock.getTime() # update time clicked until
            None
            button4_6.wasClicked = True  # if button4_6 is still clicked next frame, it is not a new click
        else:
            button4_6.wasClicked = False  # if button4_6 is clicked next frame, it is a new click
    else:
        button4_6.wasClicked = False  # if button4_6 is clicked next frame, it is a new click
    # *ISI_3* period
    if ISI_3.status == NOT_STARTED and t >= 32-frameTolerance:
        # keep track of start time/frame for later
        ISI_3.frameNStart = frameN  # exact frame index
        ISI_3.tStart = t  # local t and not account for scr refresh
        ISI_3.tStartRefresh = tThisFlipGlobal  # on global time
        win.timeOnFlip(ISI_3, 'tStartRefresh')  # time at next scr refresh
        ISI_3.start(0.1)
    elif ISI_3.status == STARTED:  # one frame should pass before updating params and completing
        ISI_3.complete()  # finish the static period
        ISI_3.tStop = ISI_3.tStart + 0.1  # record stop time

    # check for quit (typically the Esc key)
```

```
    if endExpNow or defaultKeyboard.getKeys(keyList=["escape"]):
        core.quit()

    # check if all components have finished
    if not continueRoutine:  # a component has requested a forced-end of Routine
        break
    continueRoutine = False  # will revert to True if at least one component still running
    for thisComponent in encodingComponents:
        if hasattr(thisComponent, "status") and thisComponent.status != FINISHED:
            continueRoutine = True
            break  # at least one component has not yet finished

    # refresh the screen
    if continueRoutine:  # don't flip if this routine is over or we'll get a blank screen
        win.flip()

# -------Ending Routine "encoding"-------
for thisComponent in encodingComponents:
    if hasattr(thisComponent, "setAutoDraw"):
        thisComponent.setAutoDraw(False)
encoding_go.stop()  # ensure sound has stopped at end of routine
thisExp.addData('encoding_go.started', encoding_go.tStartRefresh)
thisExp.addData('encoding_go.stopped', encoding_go.tStopRefresh)
encoding_stop.stop()  # ensure sound has stopped at end of routine
thisExp.addData('encoding_stop.started', encoding_stop.tStartRefresh)
thisExp.addData('encoding_stop.stopped', encoding_stop.tStopRefresh)

print(str(encoding_go.tStartRefresh)+";"+"encoding_go.started"+";-;-;-;-;-")
print(str(encoding_go.tStopRefresh)+";"+"encoding_go.stopped"+";-;-;-;-;-")
print(str(encoding_stop.tStartRefresh)+";"+"encoding_stop.started"+";-;-;-;-;-")
print(str(encoding_stop.tStopRefresh)+";"+"encoding_stop.stopped"+";-;-;-;-;-")

# ------Prepare to start Routine "trials_init"-------
continueRoutine = True
# update component parameters for each repeat
key_resp_4.keys = []
key_resp_4.rt = []
_key_resp_4_allKeys = []
# keep track of which components have finished
trials_initComponents = [instructions_4, key_resp_4]
for thisComponent in trials_initComponents:
    thisComponent.tStart = None
    thisComponent.tStop = None
    thisComponent.tStartRefresh = None
    thisComponent.tStopRefresh = None
    if hasattr(thisComponent, 'status'):
        thisComponent.status = NOT_STARTED
# reset timers
t = 0
_timeToFirstFrame = win.getFutureFlipTime(clock="now")
trials_initClock.reset(-_timeToFirstFrame)  # t0 is time of first possible flip
frameN = -1

# -------Run Routine "trials_init"-------
while continueRoutine:
    # get current time
    t = trials_initClock.getTime()
    tThisFlip = win.getFutureFlipTime(clock=trials_initClock)
    tThisFlipGlobal = win.getFutureFlipTime(clock=None)
```

```
    frameN = frameN + 1  # number of completed frames (so 0 is the first frame)
    # update/draw components on each frame

    # *instructions_4* updates
    if instructions_4.status == NOT_STARTED and tThisFlip >= 0.0-frameTolerance:
        # keep track of start time/frame for later
        instructions_4.frameNStart = frameN  # exact frame index
        instructions_4.tStart = t  # local t and not account for scr refresh
        instructions_4.tStartRefresh = tThisFlipGlobal  # on global time
        win.timeOnFlip(instructions_4, 'tStartRefresh')  # time at next scr refresh
        instructions_4.setAutoDraw(True)

    # *key_resp_4* updates
    waitOnFlip = False
    if key_resp_4.status == NOT_STARTED and tThisFlip >= 0.0-frameTolerance:
        # keep track of start time/frame for later
        key_resp_4.frameNStart = frameN  # exact frame index
        key_resp_4.tStart = t  # local t and not account for scr refresh
        key_resp_4.tStartRefresh = tThisFlipGlobal  # on global time
        win.timeOnFlip(key_resp_4, 'tStartRefresh')  # time at next scr refresh
        key_resp_4.status = STARTED
        # keyboard checking is just starting
        waitOnFlip = True
        win.callOnFlip(key_resp_4.clock.reset)  # t=0 on next screen flip
        win.callOnFlip(key_resp_4.clearEvents, eventType='keyboard')  # clear events on next screen
flip
    if key_resp_4.status == STARTED and not waitOnFlip:
        theseKeys = key_resp_4.getKeys(keyList=['space'], waitRelease=False)
        _key_resp_4_allKeys.extend(theseKeys)
        if len(_key_resp_4_allKeys):
            key_resp_4.keys = _key_resp_4_allKeys[-1].name  # just the last key pressed
            key_resp_4.rt = _key_resp_4_allKeys[-1].rt
            # a response ends the routine
            continueRoutine = False

    # check for quit (typically the Esc key)
    if endExpNow or defaultKeyboard.getKeys(keyList=["escape"]):
        core.quit()

    # check if all components have finished
    if not continueRoutine:  # a component has requested a forced-end of Routine
        break
    continueRoutine = False  # will revert to True if at least one component still running
    for thisComponent in trials_initComponents:
        if hasattr(thisComponent, "status") and thisComponent.status != FINISHED:
            continueRoutine = True
            break  # at least one component has not yet finished

    # refresh the screen
    if continueRoutine:  # don't flip if this routine is over or we'll get a blank screen
        win.flip()

# -------Ending Routine "trials_init"-------
for thisComponent in trials_initComponents:
    if hasattr(thisComponent, "setAutoDraw"):
        thisComponent.setAutoDraw(False)
# the Routine "trials_init" was not non-slip safe, so reset the non-slip timer
routineTimer.reset()
```

```
    # set up handler to look after randomisation of conditions etc
    trials = data.TrialHandler(nReps=40.0, method='fullRandom',
        extraInfo=expInfo, originPath=-1,
        trialList=data.importConditions('conditions.xlsx'),
        seed=None, name='trials')
    thisExp.addLoop(trials)  # add the loop to the experiment
    thisTrial = trials.trialList[0]  # so we can initialise stimuli with some values
    # abbreviate parameter names if possible (e.g. rgb = thisTrial.rgb)
    if thisTrial != None:
        for paramName in thisTrial:
            exec('{} = thisTrial[paramName]'.format(paramName))

    for thisTrial in trials:
        currentLoop = trials
        # abbreviate parameter names if possible (e.g. rgb = thisTrial.rgb)
        if thisTrial != None:
            for paramName in thisTrial:
                exec('{} = thisTrial[paramName]'.format(paramName))

        # ------Prepare to start Routine "go"-------
        continueRoutine = True
        # update component parameters for each repeat
        ready_cue.setSound('Censor Beep Sound Effect.wav', secs=1.0, hamming=True)
        ready_cue.setVolume(1.0, log=False)
        cue.setSound(sounfFilePath, secs=1.5, hamming=True)
        cue.setVolume(1.0, log=False)
        go_cue.setSound('Censor Beep Sound Effect.wav', secs=1.0, hamming=True)
        go_cue.setVolume(1.0, log=False)
        # setup some python lists for storing info about the touchgrid
        touchgrid.x = []
        touchgrid.y = []
        touchgrid.leftButton = []
        touchgrid.midButton = []
        touchgrid.rightButton = []
        touchgrid.time = []
        touchgrid.clicked_name = []
        gotValidClick = False  # until a click is received
        # keep track of which components have finished
        goComponents = [ready_cue, cue, go_cue, touchgrid, goblock1_1, goblock1_2, goblock1_3,
goblock1_4, goblock1_5, goblock1_6, goblock2_1, goblock2_2, goblock2_3, goblock2_4,
goblock2_5, goblock2_6, goblock3_1, goblock3_2, goblock3_3, goblock3_4, goblock3_5,
goblock3_6, goblock4_1, goblock4_2, goblock4_3, goblock4_4, goblock4_5, goblock4_6]
        for thisComponent in goComponents:
            thisComponent.tStart = None
            thisComponent.tStop = None
            thisComponent.tStartRefresh = None
            thisComponent.tStopRefresh = None
            if hasattr(thisComponent, 'status'):
                thisComponent.status = NOT_STARTED
        # reset timers
        t = 0
        _timeToFirstFrame = win.getFutureFlipTime(clock="now")
        goClock.reset(-_timeToFirstFrame)  # t0 is time of first possible flip
        frameN = -1

        # -------Run Routine "go"-------
        while continueRoutine:
            # get current time
            t = goClock.getTime()
```

```
    tThisFlip = win.getFutureFlipTime(clock=goClock)
    tThisFlipGlobal = win.getFutureFlipTime(clock=None)
    frameN = frameN + 1  # number of completed frames (so 0 is the first frame)
    # update/draw components on each frame
    # start/stop ready_cue
    if ready_cue.status == NOT_STARTED and tThisFlip >= 4-frameTolerance:
        # keep track of start time/frame for later
        ready_cue.frameNStart = frameN  # exact frame index
        ready_cue.tStart = t  # local t and not account for scr refresh
        ready_cue.tStartRefresh = tThisFlipGlobal  # on global time
        ready_cue.play(when=win)  # sync with win flip
    if ready_cue.status == STARTED:
        # is it time to stop? (based on global clock, using actual start)
        if tThisFlipGlobal > ready_cue.tStartRefresh + 1.0-frameTolerance:
            # keep track of stop time/frame for later
            ready_cue.tStop = t  # not accounting for scr refresh
            ready_cue.frameNStop = frameN  # exact frame index
            win.timeOnFlip(ready_cue, 'tStopRefresh')  # time at next scr refresh
            ready_cue.stop()
    # start/stop cue
    if cue.status == NOT_STARTED and tThisFlip >= 6.5-frameTolerance:
        # keep track of start time/frame for later
        cue.frameNStart = frameN  # exact frame index
        cue.tStart = t  # local t and not account for scr refresh
        cue.tStartRefresh = tThisFlipGlobal  # on global time
        cue.play(when=win)  # sync with win flip
    if cue.status == STARTED:
        # is it time to stop? (based on global clock, using actual start)
        if tThisFlipGlobal > cue.tStartRefresh + 1.5-frameTolerance:
            # keep track of stop time/frame for later
            cue.tStop = t  # not accounting for scr refresh
            cue.frameNStop = frameN  # exact frame index
            win.timeOnFlip(cue, 'tStopRefresh')  # time at next scr refresh
            cue.stop()
    # start/stop go_cue
    if go_cue.status == NOT_STARTED and tThisFlip >= 11-frameTolerance:
        # keep track of start time/frame for later
        go_cue.frameNStart = frameN  # exact frame index
        go_cue.tStart = t  # local t and not account for scr refresh
        go_cue.tStartRefresh = tThisFlipGlobal  # on global time
        go_cue.play(when=win)  # sync with win flip
    if go_cue.status == STARTED:
        # is it time to stop? (based on global clock, using actual start)
        if tThisFlipGlobal > go_cue.tStartRefresh + 1.0-frameTolerance:
            # keep track of stop time/frame for later
            go_cue.tStop = t  # not accounting for scr refresh
            go_cue.frameNStop = frameN  # exact frame index
            win.timeOnFlip(go_cue, 'tStopRefresh')  # time at next scr refresh
            go_cue.stop()
    # *touchgrid* updates
    if touchgrid.status == NOT_STARTED and tThisFlip >= 12-frameTolerance:
        # keep track of start time/frame for later
        touchgrid.frameNStart = frameN  # exact frame index
        touchgrid.tStart = t  # local t and not account for scr refresh
        touchgrid.tStartRefresh = tThisFlipGlobal  # on global time
        win.timeOnFlip(touchgrid, 'tStartRefresh')  # time at next scr refresh
        touchgrid.status = STARTED
        prevButtonState = touchgrid.getPressed()  # if button is down already this ISN'T a new click
    if touchgrid.status == STARTED:  # only update if started and not finished!
```

```
            buttons = touchgrid.getPressed()
            if buttons != prevButtonState:  # button state changed?
                prevButtonState = buttons
                if sum(buttons) > 0:  # state changed to a new click
                    # check if the mouse was inside our 'clickable' objects
                    gotValidClick = False
                    try:

iter([goblock1_1,goblock1_2,goblock1_3,goblock1_4,goblock1_5,goblock1_6,goblock2_1,goblock2_
2,goblock2_3,goblock2_4,goblock2_5,goblock2_6,goblock3_1,goblock3_2,goblock3_3,goblock3_4,
goblock3_5,goblock3_6,goblock4_1,goblock4_2,goblock4_3,goblock4_4,goblock4_5,goblock4_6])
                        clickableList =
[goblock1_1,goblock1_2,goblock1_3,goblock1_4,goblock1_5,goblock1_6,goblock2_1,goblock2_2,g
oblock2_3,goblock2_4,goblock2_5,goblock2_6,goblock3_1,goblock3_2,goblock3_3,goblock3_4,gob
lock3_5,goblock3_6,goblock4_1,goblock4_2,goblock4_3,goblock4_4,goblock4_5,goblock4_6]
                    except:
                        clickableList =
[[goblock1_1,goblock1_2,goblock1_3,goblock1_4,goblock1_5,goblock1_6,goblock2_1,goblock2_2,g
oblock2_3,goblock2_4,goblock2_5,goblock2_6,goblock3_1,goblock3_2,goblock3_3,goblock3_4,gob
lock3_5,goblock3_6,goblock4_1,goblock4_2,goblock4_3,goblock4_4,goblock4_5,goblock4_6]]
                    for obj in clickableList:
                        if obj.contains(touchgrid):
                            gotValidClick = True
                            touchgrid.clicked_name.append(obj.name)
                    x, y = touchgrid.getPos()
                    touchgrid.x.append(x)
                    touchgrid.y.append(y)
                    buttons = touchgrid.getPressed()
                    touchgrid.leftButton.append(buttons[0])
                    touchgrid.midButton.append(buttons[1])
                    touchgrid.rightButton.append(buttons[2])
                    touchgrid.time.append(globalClock.getTime())
                    # abort routine on response
                    continueRoutine = False

        # *goblock1_1* updates
        if goblock1_1.status == NOT_STARTED and tThisFlip >= 0.0-frameTolerance:
            # keep track of start time/frame for later
            goblock1_1.frameNStart = frameN  # exact frame index
            goblock1_1.tStart = t  # local t and not account for scr refresh
            goblock1_1.tStartRefresh = tThisFlipGlobal  # on global time
            win.timeOnFlip(goblock1_1, 'tStartRefresh')  # time at next scr refresh
            goblock1_1.setAutoDraw(True)

        # *goblock1_2* updates
        if goblock1_2.status == NOT_STARTED and tThisFlip >= 0.0-frameTolerance:
            # keep track of start time/frame for later
            goblock1_2.frameNStart = frameN  # exact frame index
            goblock1_2.tStart = t  # local t and not account for scr refresh
            goblock1_2.tStartRefresh = tThisFlipGlobal  # on global time
            win.timeOnFlip(goblock1_2, 'tStartRefresh')  # time at next scr refresh
            goblock1_2.setAutoDraw(True)

        # *goblock1_3* updates
        if goblock1_3.status == NOT_STARTED and tThisFlip >= 0.0-frameTolerance:
            # keep track of start time/frame for later
            goblock1_3.frameNStart = frameN  # exact frame index
            goblock1_3.tStart = t  # local t and not account for scr refresh
            goblock1_3.tStartRefresh = tThisFlipGlobal  # on global time
```

```
    win.timeOnFlip(goblock1_3, 'tStartRefresh')  # time at next scr refresh
    goblock1_3.setAutoDraw(True)

# *goblock1_4* updates
if goblock1_4.status == NOT_STARTED and tThisFlip >= 0.0-frameTolerance:
    # keep track of start time/frame for later
    goblock1_4.frameNStart = frameN  # exact frame index
    goblock1_4.tStart = t  # local t and not account for scr refresh
    goblock1_4.tStartRefresh = tThisFlipGlobal  # on global time
    win.timeOnFlip(goblock1_4, 'tStartRefresh')  # time at next scr refresh
    goblock1_4.setAutoDraw(True)

# *goblock1_5* updates
if goblock1_5.status == NOT_STARTED and tThisFlip >= 0.0-frameTolerance:
    # keep track of start time/frame for later
    goblock1_5.frameNStart = frameN  # exact frame index
    goblock1_5.tStart = t  # local t and not account for scr refresh
    goblock1_5.tStartRefresh = tThisFlipGlobal  # on global time
    win.timeOnFlip(goblock1_5, 'tStartRefresh')  # time at next scr refresh
    goblock1_5.setAutoDraw(True)

# *goblock1_6* updates
if goblock1_6.status == NOT_STARTED and tThisFlip >= 0.0-frameTolerance:
    # keep track of start time/frame for later
    goblock1_6.frameNStart = frameN  # exact frame index
    goblock1_6.tStart = t  # local t and not account for scr refresh
    goblock1_6.tStartRefresh = tThisFlipGlobal  # on global time
    win.timeOnFlip(goblock1_6, 'tStartRefresh')  # time at next scr refresh
    goblock1_6.setAutoDraw(True)

# *goblock2_1* updates
if goblock2_1.status == NOT_STARTED and tThisFlip >= 0.0-frameTolerance:
    # keep track of start time/frame for later
    goblock2_1.frameNStart = frameN  # exact frame index
    goblock2_1.tStart = t  # local t and not account for scr refresh
    goblock2_1.tStartRefresh = tThisFlipGlobal  # on global time
    win.timeOnFlip(goblock2_1, 'tStartRefresh')  # time at next scr refresh
    goblock2_1.setAutoDraw(True)

# *goblock2_2* updates
if goblock2_2.status == NOT_STARTED and tThisFlip >= 0.0-frameTolerance:
    # keep track of start time/frame for later
    goblock2_2.frameNStart = frameN  # exact frame index
    goblock2_2.tStart = t  # local t and not account for scr refresh
    goblock2_2.tStartRefresh = tThisFlipGlobal  # on global time
    win.timeOnFlip(goblock2_2, 'tStartRefresh')  # time at next scr refresh
    goblock2_2.setAutoDraw(True)

# *goblock2_3* updates
if goblock2_3.status == NOT_STARTED and tThisFlip >= 0.0-frameTolerance:
    # keep track of start time/frame for later
    goblock2_3.frameNStart = frameN  # exact frame index
    goblock2_3.tStart = t  # local t and not account for scr refresh
    goblock2_3.tStartRefresh = tThisFlipGlobal  # on global time
    win.timeOnFlip(goblock2_3, 'tStartRefresh')  # time at next scr refresh
    goblock2_3.setAutoDraw(True)

# *goblock2_4* updates
if goblock2_4.status == NOT_STARTED and tThisFlip >= 0.0-frameTolerance:
```

```
    # keep track of start time/frame for later
    goblock2_4.frameNStart = frameN  # exact frame index
    goblock2_4.tStart = t  # local t and not account for scr refresh
    goblock2_4.tStartRefresh = tThisFlipGlobal  # on global time
    win.timeOnFlip(goblock2_4, 'tStartRefresh')  # time at next scr refresh
    goblock2_4.setAutoDraw(True)

# *goblock2_5* updates
if goblock2_5.status == NOT_STARTED and tThisFlip >= 0.0-frameTolerance:
    # keep track of start time/frame for later
    goblock2_5.frameNStart = frameN  # exact frame index
    goblock2_5.tStart = t  # local t and not account for scr refresh
    goblock2_5.tStartRefresh = tThisFlipGlobal  # on global time
    win.timeOnFlip(goblock2_5, 'tStartRefresh')  # time at next scr refresh
    goblock2_5.setAutoDraw(True)

# *goblock2_6* updates
if goblock2_6.status == NOT_STARTED and tThisFlip >= 0.0-frameTolerance:
    # keep track of start time/frame for later
    goblock2_6.frameNStart = frameN  # exact frame index
    goblock2_6.tStart = t  # local t and not account for scr refresh
    goblock2_6.tStartRefresh = tThisFlipGlobal  # on global time
    win.timeOnFlip(goblock2_6, 'tStartRefresh')  # time at next scr refresh
    goblock2_6.setAutoDraw(True)

# *goblock3_1* updates
if goblock3_1.status == NOT_STARTED and tThisFlip >= 0.0-frameTolerance:
    # keep track of start time/frame for later
    goblock3_1.frameNStart = frameN  # exact frame index
    goblock3_1.tStart = t  # local t and not account for scr refresh
    goblock3_1.tStartRefresh = tThisFlipGlobal  # on global time
    win.timeOnFlip(goblock3_1, 'tStartRefresh')  # time at next scr refresh
    goblock3_1.setAutoDraw(True)

# *goblock3_2* updates
if goblock3_2.status == NOT_STARTED and tThisFlip >= 0.0-frameTolerance:
    # keep track of start time/frame for later
    goblock3_2.frameNStart = frameN  # exact frame index
    goblock3_2.tStart = t  # local t and not account for scr refresh
    goblock3_2.tStartRefresh = tThisFlipGlobal  # on global time
    win.timeOnFlip(goblock3_2, 'tStartRefresh')  # time at next scr refresh
    goblock3_2.setAutoDraw(True)

# *goblock3_3* updates
if goblock3_3.status == NOT_STARTED and tThisFlip >= 0.0-frameTolerance:
    # keep track of start time/frame for later
    goblock3_3.frameNStart = frameN  # exact frame index
    goblock3_3.tStart = t  # local t and not account for scr refresh
    goblock3_3.tStartRefresh = tThisFlipGlobal  # on global time
    win.timeOnFlip(goblock3_3, 'tStartRefresh')  # time at next scr refresh
    goblock3_3.setAutoDraw(True)

# *goblock3_4* updates
if goblock3_4.status == NOT_STARTED and tThisFlip >= 0.0-frameTolerance:
    # keep track of start time/frame for later
    goblock3_4.frameNStart = frameN  # exact frame index
    goblock3_4.tStart = t  # local t and not account for scr refresh
    goblock3_4.tStartRefresh = tThisFlipGlobal  # on global time
    win.timeOnFlip(goblock3_4, 'tStartRefresh')  # time at next scr refresh
```

```
    goblock3_4.setAutoDraw(True)

# *goblock3_5* updates
if goblock3_5.status == NOT_STARTED and tThisFlip >= 0.0-frameTolerance:
    # keep track of start time/frame for later
    goblock3_5.frameNStart = frameN  # exact frame index
    goblock3_5.tStart = t  # local t and not account for scr refresh
    goblock3_5.tStartRefresh = tThisFlipGlobal  # on global time
    win.timeOnFlip(goblock3_5, 'tStartRefresh')  # time at next scr refresh
    goblock3_5.setAutoDraw(True)

# *goblock3_6* updates
if goblock3_6.status == NOT_STARTED and tThisFlip >= 0.0-frameTolerance:
    # keep track of start time/frame for later
    goblock3_6.frameNStart = frameN  # exact frame index
    goblock3_6.tStart = t  # local t and not account for scr refresh
    goblock3_6.tStartRefresh = tThisFlipGlobal  # on global time
    win.timeOnFlip(goblock3_6, 'tStartRefresh')  # time at next scr refresh
    goblock3_6.setAutoDraw(True)

# *goblock4_1* updates
if goblock4_1.status == NOT_STARTED and tThisFlip >= 0.0-frameTolerance:
    # keep track of start time/frame for later
    goblock4_1.frameNStart = frameN  # exact frame index
    goblock4_1.tStart = t  # local t and not account for scr refresh
    goblock4_1.tStartRefresh = tThisFlipGlobal  # on global time
    win.timeOnFlip(goblock4_1, 'tStartRefresh')  # time at next scr refresh
    goblock4_1.setAutoDraw(True)

# *goblock4_2* updates
if goblock4_2.status == NOT_STARTED and tThisFlip >= 0.0-frameTolerance:
    # keep track of start time/frame for later
    goblock4_2.frameNStart = frameN  # exact frame index
    goblock4_2.tStart = t  # local t and not account for scr refresh
    goblock4_2.tStartRefresh = tThisFlipGlobal  # on global time
    win.timeOnFlip(goblock4_2, 'tStartRefresh')  # time at next scr refresh
    goblock4_2.setAutoDraw(True)

# *goblock4_3* updates
if goblock4_3.status == NOT_STARTED and tThisFlip >= 0.0-frameTolerance:
    # keep track of start time/frame for later
    goblock4_3.frameNStart = frameN  # exact frame index
    goblock4_3.tStart = t  # local t and not account for scr refresh
    goblock4_3.tStartRefresh = tThisFlipGlobal  # on global time
    win.timeOnFlip(goblock4_3, 'tStartRefresh')  # time at next scr refresh
    goblock4_3.setAutoDraw(True)

# *goblock4_4* updates
if goblock4_4.status == NOT_STARTED and tThisFlip >= 0.0-frameTolerance:
    # keep track of start time/frame for later
    goblock4_4.frameNStart = frameN  # exact frame index
    goblock4_4.tStart = t  # local t and not account for scr refresh
    goblock4_4.tStartRefresh = tThisFlipGlobal  # on global time
    win.timeOnFlip(goblock4_4, 'tStartRefresh')  # time at next scr refresh
    goblock4_4.setAutoDraw(True)

# *goblock4_5* updates
if goblock4_5.status == NOT_STARTED and tThisFlip >= 0.0-frameTolerance:
    # keep track of start time/frame for later
```

```
            goblock4_5.frameNStart = frameN  # exact frame index
            goblock4_5.tStart = t  # local t and not account for scr refresh
            goblock4_5.tStartRefresh = tThisFlipGlobal  # on global time
            win.timeOnFlip(goblock4_5, 'tStartRefresh')  # time at next scr refresh
            goblock4_5.setAutoDraw(True)

        # *goblock4_6* updates
        if goblock4_6.status == NOT_STARTED and tThisFlip >= 0.0-frameTolerance:
            # keep track of start time/frame for later
            goblock4_6.frameNStart = frameN  # exact frame index
            goblock4_6.tStart = t  # local t and not account for scr refresh
            goblock4_6.tStartRefresh = tThisFlipGlobal  # on global time
            win.timeOnFlip(goblock4_6, 'tStartRefresh')  # time at next scr refresh
            goblock4_6.setAutoDraw(True)

        # check for quit (typically the Esc key)
        if endExpNow or defaultKeyboard.getKeys(keyList=["escape"]):
            core.quit()

        # check if all components have finished
        if not continueRoutine:  # a component has requested a forced-end of Routine
            break
        continueRoutine = False  # will revert to True if at least one component still running
        for thisComponent in goComponents:
            if hasattr(thisComponent, "status") and thisComponent.status != FINISHED:
                continueRoutine = True
                break  # at least one component has not yet finished

        # refresh the screen
        if continueRoutine:  # don't flip if this routine is over or we'll get a blank screen
            win.flip()

    # -------Ending Routine "go"-------
    for thisComponent in goComponents:
        if hasattr(thisComponent, "setAutoDraw"):
            thisComponent.setAutoDraw(False)
    ready_cue.stop()  # ensure sound has stopped at end of routine
    trials.addData('ready_cue.started', ready_cue.tStartRefresh)
    trials.addData('ready_cue.stopped', ready_cue.tStopRefresh)
    cue.stop()  # ensure sound has stopped at end of routine
    trials.addData('cue.started', cue.tStartRefresh)
    trials.addData('cue.stopped', cue.tStopRefresh)
    go_cue.stop()  # ensure sound has stopped at end of routine
    trials.addData('go_cue.started', go_cue.tStartRefresh)
    trials.addData('go_cue.stopped', go_cue.tStopRefresh)
    # store data for trials (TrialHandler)
    if len(touchgrid.x): trials.addData('touchgrid.x', touchgrid.x[0])
    if len(touchgrid.y): trials.addData('touchgrid.y', touchgrid.y[0])
    #if len(touchgrid.leftButton): trials.addData('touchgrid.leftButton', touchgrid.leftButton[0])
    #if len(touchgrid.midButton): trials.addData('touchgrid.midButton', touchgrid.midButton[0])
    #if len(touchgrid.rightButton): trials.addData('touchgrid.rightButton', touchgrid.rightButton[0])
    if len(touchgrid.time): trials.addData('touchgrid.time', touchgrid.time[0])
    if len(touchgrid.clicked_name): trials.addData('touchgrid.clicked_name',
touchgrid.clicked_name[0])
    trials.addData('touchgrid.started', touchgrid.tStartRefresh)
    #trials.addData('touchgrid.stopped', touchgrid.tStopRefresh)

    print(str(ready_cue.tStartRefresh)+";"+"ready_cue.started"+";"+str(trials.thisTrialN)+";-;-;-;-")
    print(str(ready_cue.tStopRefresh)+";"+"ready_cue.stopped"+";"+str(trials.thisTrialN)+";-;-;-;-")
```

```
    print(str(cue.tStartRefresh)+";"+"cue.started"+";"+str(trials.thisTrialN)+";-
;"+thisTrial.sounfFilePath+";-;-")
    print(str(cue.tStopRefresh)+";"+"cue.stopped"+";"+str(trials.thisTrialN)+";-;-;-;-")
    print(str(go_cue.tStartRefresh)+";"+"go_cue.started"+";"+str(trials.thisTrialN)+";-
;"+thisTrial.corrAns+";-;-")
    print(str(go_cue.tStopRefresh)+";"+"go_cue.stopped"+";"+str(trials.thisTrialN)+";-;-;-;-")
    #print(str(encoding_go.tStartRefresh)+";"+"touchgrid"+";"+str(trials.thisTrialN)+";-;-;-;-")
    if len(touchgrid.time) and len(touchgrid.clicked_name):

print(str(touchgrid.time[0])+";"+"touchgrid.clicked"+";"+str(trials.thisTrialN)+";"+touchgrid.clicked_
name[0]+";"+thisTrial.corrAns+";"+str(touchgrid.x[0])+";"+str(touchgrid.y[0]))
    elif len(touchgrid.time):
        print(str(touchgrid.time[0])+";"+"touchgrid.clicked"+";"+str(trials.thisTrialN)+";"+"-
"+";"+thisTrial.corrAns+";"+str(touchgrid.x[0])+";"+str(touchgrid.y[0]))
    else:
        print(str(touchgrid.time[0])+";"+"touchgrid.notclicked"+";"+str(trials.thisTrialN)+";"+"-
"+";"+thisTrial.corrAns+";"+"-"+";"+"-")

    # the Routine "go" was not non-slip safe, so reset the non-slip timer
    routineTimer.reset()

    # ------Prepare to start Routine "back"-------
    continueRoutine = True
    # update component parameters for each repeat
    return_cue.setSound('Censor Beep Sound Effect.wav', secs=1.0, hamming=True)
    return_cue.setVolume(1.0, log=False)
    trial_end.keys = []
    trial_end.rt = []
    _trial_end_allKeys = []
    # keep track of which components have finished
    backComponents = [return_cue, trial_end, backgrid1_1, backgrid1_2, backgrid1_3,
backgrid1_4, backgrid1_5, backgrid1_6, backgrid2_1, backgrid2_2, backgrid2_3, backgrid2_4,
backgrid2_5, backgrid2_6, backgrid3_1, backgrid3_2, backgrid3_3, backgrid3_4, backgrid3_5,
backgrid3_6, backgrid4_1, backgrid4_2, backgrid4_3, backgrid4_4, backgrid4_5, backgrid4_6]
    for thisComponent in backComponents:
        thisComponent.tStart = None
        thisComponent.tStop = None
        thisComponent.tStartRefresh = None
        thisComponent.tStopRefresh = None
        if hasattr(thisComponent, 'status'):
            thisComponent.status = NOT_STARTED
    # reset timers
    t = 0
    _timeToFirstFrame = win.getFutureFlipTime(clock="now")
    backClock.reset(-_timeToFirstFrame)  # t0 is time of first possible flip
    frameN = -1

    # -------Run Routine "back"-------
    while continueRoutine:
        # get current time
        t = backClock.getTime()
        tThisFlip = win.getFutureFlipTime(clock=backClock)
        tThisFlipGlobal = win.getFutureFlipTime(clock=None)
        frameN = frameN + 1  # number of completed frames (so 0 is the first frame)
        # update/draw components on each frame
        # start/stop return_cue
        if return_cue.status == NOT_STARTED and tThisFlip >= 0-frameTolerance:
            # keep track of start time/frame for later
            return_cue.frameNStart = frameN  # exact frame index
```

```
            return_cue.tStart = t  # local t and not account for scr refresh
            return_cue.tStartRefresh = tThisFlipGlobal  # on global time
            return_cue.play(when=win)  # sync with win flip
        if return_cue.status == STARTED:
            # is it time to stop? (based on global clock, using actual start)
            if tThisFlipGlobal > return_cue.tStartRefresh + 1.0-frameTolerance:
                # keep track of stop time/frame for later
                return_cue.tStop = t  # not accounting for scr refresh
                return_cue.frameNStop = frameN  # exact frame index
                win.timeOnFlip(return_cue, 'tStopRefresh')  # time at next scr refresh
                return_cue.stop()

        # *trial_end* updates
        waitOnFlip = False
        if trial_end.status == NOT_STARTED and tThisFlip >= 1-frameTolerance:
            # keep track of start time/frame for later
            trial_end.frameNStart = frameN  # exact frame index
            trial_end.tStart = t  # local t and not account for scr refresh
            trial_end.tStartRefresh = tThisFlipGlobal  # on global time
            win.timeOnFlip(trial_end, 'tStartRefresh')  # time at next scr refresh
            trial_end.status = STARTED
            # keyboard checking is just starting
            waitOnFlip = True
            win.callOnFlip(trial_end.clock.reset)  # t=0 on next screen flip
        if trial_end.status == STARTED and not waitOnFlip:
            theseKeys = trial_end.getKeys(keyList=['space'], waitRelease=False)
            _trial_end_allKeys.extend(theseKeys)
            if len(_trial_end_allKeys):
                trial_end.keys = _trial_end_allKeys[-1].name  # just the last key pressed
                trial_end.rt = _trial_end_allKeys[-1].rt
                # was this correct?
                if (trial_end.keys == str(corrAns)) or (trial_end.keys == corrAns):
                    trial_end.corr = 1
                else:
                    trial_end.corr = 0
                # a response ends the routine
                continueRoutine = False

        # *backgrid1_1* updates
        if backgrid1_1.status == NOT_STARTED and tThisFlip >= 1-frameTolerance:
            # keep track of start time/frame for later
            backgrid1_1.frameNStart = frameN  # exact frame index
            backgrid1_1.tStart = t  # local t and not account for scr refresh
            backgrid1_1.tStartRefresh = tThisFlipGlobal  # on global time
            win.timeOnFlip(backgrid1_1, 'tStartRefresh')  # time at next scr refresh
            backgrid1_1.setAutoDraw(True)
        if backgrid1_1.status == STARTED:
            # check whether backgrid1_1 has been pressed
            if backgrid1_1.isClicked:
                if not backgrid1_1.wasClicked:
                    backgrid1_1.timesOn.append(globalClock.getTime()) # store time of first click
                    backgrid1_1.timesOff.append(globalClock.getTime()) # store time clicked until
                else:
                    backgrid1_1.timesOff[-1] = globalClock.getTime() # update time clicked until
                None
                backgrid1_1.wasClicked = True  # if backgrid1_1 is still clicked next frame, it is not a new
click
            else:
                backgrid1_1.wasClicked = False  # if backgrid1_1 is clicked next frame, it is a new click
```

```
        else:
            backgrid1_1.wasClicked = False  # if backgrid1_1 is clicked next frame, it is a new click

        # *backgrid1_2* updates
        if backgrid1_2.status == NOT_STARTED and tThisFlip >= 1-frameTolerance:
            # keep track of start time/frame for later
            backgrid1_2.frameNStart = frameN  # exact frame index
            backgrid1_2.tStart = t  # local t and not account for scr refresh
            backgrid1_2.tStartRefresh = tThisFlipGlobal  # on global time
            win.timeOnFlip(backgrid1_2, 'tStartRefresh')  # time at next scr refresh
            backgrid1_2.setAutoDraw(True)
        if backgrid1_2.status == STARTED:
            # check whether backgrid1_2 has been pressed
            if backgrid1_2.isClicked:
                if not backgrid1_2.wasClicked:
                    backgrid1_2.timesOn.append(globalClock.getTime()) # store time of first click
                    backgrid1_2.timesOff.append(globalClock.getTime()) # store time clicked until
                else:
                    backgrid1_2.timesOff[-1] = globalClock.getTime() # update time clicked until
                None
                backgrid1_2.wasClicked = True  # if backgrid1_2 is still clicked next frame, it is not a new
click
            else:
                backgrid1_2.wasClicked = False  # if backgrid1_2 is clicked next frame, it is a new click
        else:
            backgrid1_2.wasClicked = False  # if backgrid1_2 is clicked next frame, it is a new click

        # *backgrid1_3* updates
        if backgrid1_3.status == NOT_STARTED and tThisFlip >= 1-frameTolerance:
            # keep track of start time/frame for later
            backgrid1_3.frameNStart = frameN  # exact frame index
            backgrid1_3.tStart = t  # local t and not account for scr refresh
            backgrid1_3.tStartRefresh = tThisFlipGlobal  # on global time
            win.timeOnFlip(backgrid1_3, 'tStartRefresh')  # time at next scr refresh
            backgrid1_3.setAutoDraw(True)
        if backgrid1_3.status == STARTED:
            # check whether backgrid1_3 has been pressed
            if backgrid1_3.isClicked:
                if not backgrid1_3.wasClicked:
                    backgrid1_3.timesOn.append(globalClock.getTime()) # store time of first click
                    backgrid1_3.timesOff.append(globalClock.getTime()) # store time clicked until
                else:
                    backgrid1_3.timesOff[-1] = globalClock.getTime() # update time clicked until
                None
                backgrid1_3.wasClicked = True  # if backgrid1_3 is still clicked next frame, it is not a new
click
            else:
                backgrid1_3.wasClicked = False  # if backgrid1_3 is clicked next frame, it is a new click
        else:
            backgrid1_3.wasClicked = False  # if backgrid1_3 is clicked next frame, it is a new click

        # *backgrid1_4* updates
        if backgrid1_4.status == NOT_STARTED and tThisFlip >= 1-frameTolerance:
            # keep track of start time/frame for later
            backgrid1_4.frameNStart = frameN  # exact frame index
            backgrid1_4.tStart = t  # local t and not account for scr refresh
            backgrid1_4.tStartRefresh = tThisFlipGlobal  # on global time
            win.timeOnFlip(backgrid1_4, 'tStartRefresh')  # time at next scr refresh
            backgrid1_4.setAutoDraw(True)
```

```
        if backgrid1_4.status == STARTED:
            # check whether backgrid1_4 has been pressed
            if backgrid1_4.isClicked:
                if not backgrid1_4.wasClicked:
                    backgrid1_4.timesOn.append(globalClock.getTime()) # store time of first click
                    backgrid1_4.timesOff.append(globalClock.getTime()) # store time clicked until
                else:
                    backgrid1_4.timesOff[-1] = globalClock.getTime() # update time clicked until
                None
                backgrid1_4.wasClicked = True  # if backgrid1_4 is still clicked next frame, it is not a new
click
            else:
                backgrid1_4.wasClicked = False  # if backgrid1_4 is clicked next frame, it is a new click
        else:
            backgrid1_4.wasClicked = False  # if backgrid1_4 is clicked next frame, it is a new click

        # *backgrid1_5* updates
        if backgrid1_5.status == NOT_STARTED and tThisFlip >= 1-frameTolerance:
            # keep track of start time/frame for later
            backgrid1_5.frameNStart = frameN  # exact frame index
            backgrid1_5.tStart = t  # local t and not account for scr refresh
            backgrid1_5.tStartRefresh = tThisFlipGlobal  # on global time
            win.timeOnFlip(backgrid1_5, 'tStartRefresh')  # time at next scr refresh
            backgrid1_5.setAutoDraw(True)
        if backgrid1_5.status == STARTED:
            # check whether backgrid1_5 has been pressed
            if backgrid1_5.isClicked:
                if not backgrid1_5.wasClicked:
                    backgrid1_5.timesOn.append(globalClock.getTime()) # store time of first click
                    backgrid1_5.timesOff.append(globalClock.getTime()) # store time clicked until
                else:
                    backgrid1_5.timesOff[-1] = globalClock.getTime() # update time clicked until
                None
                backgrid1_5.wasClicked = True  # if backgrid1_5 is still clicked next frame, it is not a new
click
            else:
                backgrid1_5.wasClicked = False  # if backgrid1_5 is clicked next frame, it is a new click
        else:
            backgrid1_5.wasClicked = False  # if backgrid1_5 is clicked next frame, it is a new click

        # *backgrid1_6* updates
        if backgrid1_6.status == NOT_STARTED and tThisFlip >= 1-frameTolerance:
            # keep track of start time/frame for later
            backgrid1_6.frameNStart = frameN  # exact frame index
            backgrid1_6.tStart = t  # local t and not account for scr refresh
            backgrid1_6.tStartRefresh = tThisFlipGlobal  # on global time
            win.timeOnFlip(backgrid1_6, 'tStartRefresh')  # time at next scr refresh
            backgrid1_6.setAutoDraw(True)
        if backgrid1_6.status == STARTED:
            # check whether backgrid1_6 has been pressed
            if backgrid1_6.isClicked:
                if not backgrid1_6.wasClicked:
                    backgrid1_6.timesOn.append(globalClock.getTime()) # store time of first click
                    backgrid1_6.timesOff.append(globalClock.getTime()) # store time clicked until
                else:
                    backgrid1_6.timesOff[-1] = globalClock.getTime() # update time clicked until
                None
                backgrid1_6.wasClicked = True  # if backgrid1_6 is still clicked next frame, it is not a new
click
```

```
            else:
                backgrid1_6.wasClicked = False  # if backgrid1_6 is clicked next frame, it is a new click
        else:
            backgrid1_6.wasClicked = False  # if backgrid1_6 is clicked next frame, it is a new click

        # *backgrid2_1* updates
        if backgrid2_1.status == NOT_STARTED and tThisFlip >= 1-frameTolerance:
            # keep track of start time/frame for later
            backgrid2_1.frameNStart = frameN  # exact frame index
            backgrid2_1.tStart = t  # local t and not account for scr refresh
            backgrid2_1.tStartRefresh = tThisFlipGlobal  # on global time
            win.timeOnFlip(backgrid2_1, 'tStartRefresh')  # time at next scr refresh
            backgrid2_1.setAutoDraw(True)
        if backgrid2_1.status == STARTED:
            # check whether backgrid2_1 has been pressed
            if backgrid2_1.isClicked:
                if not backgrid2_1.wasClicked:
                    backgrid2_1.timesOn.append(globalClock.getTime()) # store time of first click
                    backgrid2_1.timesOff.append(globalClock.getTime()) # store time clicked until
                else:
                    backgrid2_1.timesOff[-1] = globalClock.getTime() # update time clicked until
                None
                backgrid2_1.wasClicked = True  # if backgrid2_1 is still clicked next frame, it is not a new
click
            else:
                backgrid2_1.wasClicked = False  # if backgrid2_1 is clicked next frame, it is a new click
        else:
            backgrid2_1.wasClicked = False  # if backgrid2_1 is clicked next frame, it is a new click

        # *backgrid2_2* updates
        if backgrid2_2.status == NOT_STARTED and tThisFlip >= 1-frameTolerance:
            # keep track of start time/frame for later
            backgrid2_2.frameNStart = frameN  # exact frame index
            backgrid2_2.tStart = t  # local t and not account for scr refresh
            backgrid2_2.tStartRefresh = tThisFlipGlobal  # on global time
            win.timeOnFlip(backgrid2_2, 'tStartRefresh')  # time at next scr refresh
            backgrid2_2.setAutoDraw(True)
        if backgrid2_2.status == STARTED:
            # check whether backgrid2_2 has been pressed
            if backgrid2_2.isClicked:
                if not backgrid2_2.wasClicked:
                    backgrid2_2.timesOn.append(globalClock.getTime()) # store time of first click
                    backgrid2_2.timesOff.append(globalClock.getTime()) # store time clicked until
                else:
                    backgrid2_2.timesOff[-1] = globalClock.getTime() # update time clicked until
                None
                backgrid2_2.wasClicked = True  # if backgrid2_2 is still clicked next frame, it is not a new
click
            else:
                backgrid2_2.wasClicked = False  # if backgrid2_2 is clicked next frame, it is a new click
        else:
            backgrid2_2.wasClicked = False  # if backgrid2_2 is clicked next frame, it is a new click

        # *backgrid2_3* updates
        if backgrid2_3.status == NOT_STARTED and tThisFlip >= 1-frameTolerance:
            # keep track of start time/frame for later
            backgrid2_3.frameNStart = frameN  # exact frame index
            backgrid2_3.tStart = t  # local t and not account for scr refresh
            backgrid2_3.tStartRefresh = tThisFlipGlobal  # on global time
```

```
            win.timeOnFlip(backgrid2_3, 'tStartRefresh')  # time at next scr refresh
            backgrid2_3.setAutoDraw(True)
        if backgrid2_3.status == STARTED:
            # check whether backgrid2_3 has been pressed
            if backgrid2_3.isClicked:
                if not backgrid2_3.wasClicked:
                    backgrid2_3.timesOn.append(globalClock.getTime()) # store time of first click
                    backgrid2_3.timesOff.append(globalClock.getTime()) # store time clicked until
                else:
                    backgrid2_3.timesOff[-1] = globalClock.getTime() # update time clicked until
                None
                backgrid2_3.wasClicked = True  # if backgrid2_3 is still clicked next frame, it is not a new
click
            else:
                backgrid2_3.wasClicked = False  # if backgrid2_3 is clicked next frame, it is a new click
        else:
            backgrid2_3.wasClicked = False  # if backgrid2_3 is clicked next frame, it is a new click

        # *backgrid2_4* updates
        if backgrid2_4.status == NOT_STARTED and tThisFlip >= 1-frameTolerance:
            # keep track of start time/frame for later
            backgrid2_4.frameNStart = frameN  # exact frame index
            backgrid2_4.tStart = t  # local t and not account for scr refresh
            backgrid2_4.tStartRefresh = tThisFlipGlobal  # on global time
            win.timeOnFlip(backgrid2_4, 'tStartRefresh')  # time at next scr refresh
            backgrid2_4.setAutoDraw(True)
        if backgrid2_4.status == STARTED:
            # check whether backgrid2_4 has been pressed
            if backgrid2_4.isClicked:
                if not backgrid2_4.wasClicked:
                    backgrid2_4.timesOn.append(globalClock.getTime()) # store time of first click
                    backgrid2_4.timesOff.append(globalClock.getTime()) # store time clicked until
                else:
                    backgrid2_4.timesOff[-1] = globalClock.getTime() # update time clicked until
                None
                backgrid2_4.wasClicked = True  # if backgrid2_4 is still clicked next frame, it is not a new
click
            else:
                backgrid2_4.wasClicked = False  # if backgrid2_4 is clicked next frame, it is a new click
        else:
            backgrid2_4.wasClicked = False  # if backgrid2_4 is clicked next frame, it is a new click

        # *backgrid2_5* updates
        if backgrid2_5.status == NOT_STARTED and tThisFlip >= 1-frameTolerance:
            # keep track of start time/frame for later
            backgrid2_5.frameNStart = frameN  # exact frame index
            backgrid2_5.tStart = t  # local t and not account for scr refresh
            backgrid2_5.tStartRefresh = tThisFlipGlobal  # on global time
            win.timeOnFlip(backgrid2_5, 'tStartRefresh')  # time at next scr refresh
            backgrid2_5.setAutoDraw(True)
        if backgrid2_5.status == STARTED:
            # check whether backgrid2_5 has been pressed
            if backgrid2_5.isClicked:
                if not backgrid2_5.wasClicked:
                    backgrid2_5.timesOn.append(globalClock.getTime()) # store time of first click
                    backgrid2_5.timesOff.append(globalClock.getTime()) # store time clicked until
                else:
                    backgrid2_5.timesOff[-1] = globalClock.getTime() # update time clicked until
                None
```

```
            backgrid2_5.wasClicked = True  # if backgrid2_5 is still clicked next frame, it is not a new
click
          else:
            backgrid2_5.wasClicked = False  # if backgrid2_5 is clicked next frame, it is a new click
        else:
          backgrid2_5.wasClicked = False  # if backgrid2_5 is clicked next frame, it is a new click

        # *backgrid2_6* updates
        if backgrid2_6.status == NOT_STARTED and tThisFlip >= 1-frameTolerance:
          # keep track of start time/frame for later
          backgrid2_6.frameNStart = frameN  # exact frame index
          backgrid2_6.tStart = t  # local t and not account for scr refresh
          backgrid2_6.tStartRefresh = tThisFlipGlobal  # on global time
          win.timeOnFlip(backgrid2_6, 'tStartRefresh')  # time at next scr refresh
          backgrid2_6.setAutoDraw(True)
        if backgrid2_6.status == STARTED:
          # check whether backgrid2_6 has been pressed
          if backgrid2_6.isClicked:
            if not backgrid2_6.wasClicked:
              backgrid2_6.timesOn.append(globalClock.getTime()) # store time of first click
              backgrid2_6.timesOff.append(globalClock.getTime()) # store time clicked until
            else:
              backgrid2_6.timesOff[-1] = globalClock.getTime() # update time clicked until
            None
            backgrid2_6.wasClicked = True  # if backgrid2_6 is still clicked next frame, it is not a new
click
          else:
            backgrid2_6.wasClicked = False  # if backgrid2_6 is clicked next frame, it is a new click
        else:
          backgrid2_6.wasClicked = False  # if backgrid2_6 is clicked next frame, it is a new click

        # *backgrid3_1* updates
        if backgrid3_1.status == NOT_STARTED and tThisFlip >= 1-frameTolerance:
          # keep track of start time/frame for later
          backgrid3_1.frameNStart = frameN  # exact frame index
          backgrid3_1.tStart = t  # local t and not account for scr refresh
          backgrid3_1.tStartRefresh = tThisFlipGlobal  # on global time
          win.timeOnFlip(backgrid3_1, 'tStartRefresh')  # time at next scr refresh
          backgrid3_1.setAutoDraw(True)
        if backgrid3_1.status == STARTED:
          # check whether backgrid3_1 has been pressed
          if backgrid3_1.isClicked:
            if not backgrid3_1.wasClicked:
              backgrid3_1.timesOn.append(globalClock.getTime()) # store time of first click
              backgrid3_1.timesOff.append(globalClock.getTime()) # store time clicked until
            else:
              backgrid3_1.timesOff[-1] = globalClock.getTime() # update time clicked until
            None
            backgrid3_1.wasClicked = True  # if backgrid3_1 is still clicked next frame, it is not a new
click
          else:
            backgrid3_1.wasClicked = False  # if backgrid3_1 is clicked next frame, it is a new click
        else:
          backgrid3_1.wasClicked = False  # if backgrid3_1 is clicked next frame, it is a new click

        # *backgrid3_2* updates
        if backgrid3_2.status == NOT_STARTED and tThisFlip >= 1-frameTolerance:
          # keep track of start time/frame for later
          backgrid3_2.frameNStart = frameN  # exact frame index
```

```
            backgrid3_2.tStart = t  # local t and not account for scr refresh
            backgrid3_2.tStartRefresh = tThisFlipGlobal  # on global time
            win.timeOnFlip(backgrid3_2, 'tStartRefresh')  # time at next scr refresh
            backgrid3_2.setAutoDraw(True)
        if backgrid3_2.status == STARTED:
            # check whether backgrid3_2 has been pressed
            if backgrid3_2.isClicked:
                if not backgrid3_2.wasClicked:
                    backgrid3_2.timesOn.append(globalClock.getTime()) # store time of first click
                    backgrid3_2.timesOff.append(globalClock.getTime()) # store time clicked until
                else:
                    backgrid3_2.timesOff[-1] = globalClock.getTime() # update time clicked until
                None
                backgrid3_2.wasClicked = True  # if backgrid3_2 is still clicked next frame, it is not a new
click
            else:
                backgrid3_2.wasClicked = False  # if backgrid3_2 is clicked next frame, it is a new click
        else:
            backgrid3_2.wasClicked = False  # if backgrid3_2 is clicked next frame, it is a new click

        # *backgrid3_3* updates
        if backgrid3_3.status == NOT_STARTED and tThisFlip >= 1-frameTolerance:
            # keep track of start time/frame for later
            backgrid3_3.frameNStart = frameN  # exact frame index
            backgrid3_3.tStart = t  # local t and not account for scr refresh
            backgrid3_3.tStartRefresh = tThisFlipGlobal  # on global time
            win.timeOnFlip(backgrid3_3, 'tStartRefresh')  # time at next scr refresh
            backgrid3_3.setAutoDraw(True)
        if backgrid3_3.status == STARTED:
            # check whether backgrid3_3 has been pressed
            if backgrid3_3.isClicked:
                if not backgrid3_3.wasClicked:
                    backgrid3_3.timesOn.append(globalClock.getTime()) # store time of first click
                    backgrid3_3.timesOff.append(globalClock.getTime()) # store time clicked until
                else:
                    backgrid3_3.timesOff[-1] = globalClock.getTime() # update time clicked until
                None
                backgrid3_3.wasClicked = True  # if backgrid3_3 is still clicked next frame, it is not a new
click
            else:
                backgrid3_3.wasClicked = False  # if backgrid3_3 is clicked next frame, it is a new click
        else:
            backgrid3_3.wasClicked = False  # if backgrid3_3 is clicked next frame, it is a new click

        # *backgrid3_4* updates
        if backgrid3_4.status == NOT_STARTED and tThisFlip >= 1-frameTolerance:
            # keep track of start time/frame for later
            backgrid3_4.frameNStart = frameN  # exact frame index
            backgrid3_4.tStart = t  # local t and not account for scr refresh
            backgrid3_4.tStartRefresh = tThisFlipGlobal  # on global time
            win.timeOnFlip(backgrid3_4, 'tStartRefresh')  # time at next scr refresh
            backgrid3_4.setAutoDraw(True)
        if backgrid3_4.status == STARTED:
            # check whether backgrid3_4 has been pressed
            if backgrid3_4.isClicked:
                if not backgrid3_4.wasClicked:
                    backgrid3_4.timesOn.append(globalClock.getTime()) # store time of first click
                    backgrid3_4.timesOff.append(globalClock.getTime()) # store time clicked until
                else:
```

```
                backgrid3_4.timesOff[-1] = globalClock.getTime() # update time clicked until
            None
            backgrid3_4.wasClicked = True  # if backgrid3_4 is still clicked next frame, it is not a new
click
          else:
            backgrid3_4.wasClicked = False  # if backgrid3_4 is clicked next frame, it is a new click
        else:
          backgrid3_4.wasClicked = False  # if backgrid3_4 is clicked next frame, it is a new click

        # *backgrid3_5* updates
        if backgrid3_5.status == NOT_STARTED and tThisFlip >= 1-frameTolerance:
          # keep track of start time/frame for later
          backgrid3_5.frameNStart = frameN  # exact frame index
          backgrid3_5.tStart = t  # local t and not account for scr refresh
          backgrid3_5.tStartRefresh = tThisFlipGlobal  # on global time
          win.timeOnFlip(backgrid3_5, 'tStartRefresh')  # time at next scr refresh
          backgrid3_5.setAutoDraw(True)
        if backgrid3_5.status == STARTED:
          # check whether backgrid3_5 has been pressed
          if backgrid3_5.isClicked:
            if not backgrid3_5.wasClicked:
              backgrid3_5.timesOn.append(globalClock.getTime()) # store time of first click
              backgrid3_5.timesOff.append(globalClock.getTime()) # store time clicked until
            else:
              backgrid3_5.timesOff[-1] = globalClock.getTime() # update time clicked until
            None
            backgrid3_5.wasClicked = True  # if backgrid3_5 is still clicked next frame, it is not a new
click
          else:
            backgrid3_5.wasClicked = False  # if backgrid3_5 is clicked next frame, it is a new click
        else:
          backgrid3_5.wasClicked = False  # if backgrid3_5 is clicked next frame, it is a new click

        # *backgrid3_6* updates
        if backgrid3_6.status == NOT_STARTED and tThisFlip >= 1-frameTolerance:
          # keep track of start time/frame for later
          backgrid3_6.frameNStart = frameN  # exact frame index
          backgrid3_6.tStart = t  # local t and not account for scr refresh
          backgrid3_6.tStartRefresh = tThisFlipGlobal  # on global time
          win.timeOnFlip(backgrid3_6, 'tStartRefresh')  # time at next scr refresh
          backgrid3_6.setAutoDraw(True)
        if backgrid3_6.status == STARTED:
          # check whether backgrid3_6 has been pressed
          if backgrid3_6.isClicked:
            if not backgrid3_6.wasClicked:
              backgrid3_6.timesOn.append(globalClock.getTime()) # store time of first click
              backgrid3_6.timesOff.append(globalClock.getTime()) # store time clicked until
            else:
              backgrid3_6.timesOff[-1] = globalClock.getTime() # update time clicked until
            None
            backgrid3_6.wasClicked = True  # if backgrid3_6 is still clicked next frame, it is not a new
click
          else:
            backgrid3_6.wasClicked = False  # if backgrid3_6 is clicked next frame, it is a new click
        else:
          backgrid3_6.wasClicked = False  # if backgrid3_6 is clicked next frame, it is a new click

        # *backgrid4_1* updates
        if backgrid4_1.status == NOT_STARTED and tThisFlip >= 1-frameTolerance:
```

```
            # keep track of start time/frame for later
            backgrid4_1.frameNStart = frameN  # exact frame index
            backgrid4_1.tStart = t  # local t and not account for scr refresh
            backgrid4_1.tStartRefresh = tThisFlipGlobal  # on global time
            win.timeOnFlip(backgrid4_1, 'tStartRefresh')  # time at next scr refresh
            backgrid4_1.setAutoDraw(True)
        if backgrid4_1.status == STARTED:
            # check whether backgrid4_1 has been pressed
            if backgrid4_1.isClicked:
                if not backgrid4_1.wasClicked:
                    backgrid4_1.timesOn.append(globalClock.getTime()) # store time of first click
                    backgrid4_1.timesOff.append(globalClock.getTime()) # store time clicked until
                else:
                    backgrid4_1.timesOff[-1] = globalClock.getTime() # update time clicked until
                None
                backgrid4_1.wasClicked = True  # if backgrid4_1 is still clicked next frame, it is not a new
click
            else:
                backgrid4_1.wasClicked = False  # if backgrid4_1 is clicked next frame, it is a new click
        else:
            backgrid4_1.wasClicked = False  # if backgrid4_1 is clicked next frame, it is a new click

        # *backgrid4_2* updates
        if backgrid4_2.status == NOT_STARTED and tThisFlip >= 1-frameTolerance:
            # keep track of start time/frame for later
            backgrid4_2.frameNStart = frameN  # exact frame index
            backgrid4_2.tStart = t  # local t and not account for scr refresh
            backgrid4_2.tStartRefresh = tThisFlipGlobal  # on global time
            win.timeOnFlip(backgrid4_2, 'tStartRefresh')  # time at next scr refresh
            backgrid4_2.setAutoDraw(True)
        if backgrid4_2.status == STARTED:
            # check whether backgrid4_2 has been pressed
            if backgrid4_2.isClicked:
                if not backgrid4_2.wasClicked:
                    backgrid4_2.timesOn.append(globalClock.getTime()) # store time of first click
                    backgrid4_2.timesOff.append(globalClock.getTime()) # store time clicked until
                else:
                    backgrid4_2.timesOff[-1] = globalClock.getTime() # update time clicked until
                None
                backgrid4_2.wasClicked = True  # if backgrid4_2 is still clicked next frame, it is not a new
click
            else:
                backgrid4_2.wasClicked = False  # if backgrid4_2 is clicked next frame, it is a new click
        else:
            backgrid4_2.wasClicked = False  # if backgrid4_2 is clicked next frame, it is a new click

        # *backgrid4_3* updates
        if backgrid4_3.status == NOT_STARTED and tThisFlip >= 1-frameTolerance:
            # keep track of start time/frame for later
            backgrid4_3.frameNStart = frameN  # exact frame index
            backgrid4_3.tStart = t  # local t and not account for scr refresh
            backgrid4_3.tStartRefresh = tThisFlipGlobal  # on global time
            win.timeOnFlip(backgrid4_3, 'tStartRefresh')  # time at next scr refresh
            backgrid4_3.setAutoDraw(True)
        if backgrid4_3.status == STARTED:
            # check whether backgrid4_3 has been pressed
            if backgrid4_3.isClicked:
                if not backgrid4_3.wasClicked:
                    backgrid4_3.timesOn.append(globalClock.getTime()) # store time of first click
```

```
            backgrid4_3.timesOff.append(globalClock.getTime()) # store time clicked until
          else:
            backgrid4_3.timesOff[-1] = globalClock.getTime() # update time clicked until
          None
          backgrid4_3.wasClicked = True  # if backgrid4_3 is still clicked next frame, it is not a new
click
        else:
          backgrid4_3.wasClicked = False  # if backgrid4_3 is clicked next frame, it is a new click
      else:
        backgrid4_3.wasClicked = False  # if backgrid4_3 is clicked next frame, it is a new click

      # *backgrid4_4* updates
      if backgrid4_4.status == NOT_STARTED and tThisFlip >= 1-frameTolerance:
        # keep track of start time/frame for later
        backgrid4_4.frameNStart = frameN  # exact frame index
        backgrid4_4.tStart = t  # local t and not account for scr refresh
        backgrid4_4.tStartRefresh = tThisFlipGlobal  # on global time
        win.timeOnFlip(backgrid4_4, 'tStartRefresh')  # time at next scr refresh
        backgrid4_4.setAutoDraw(True)
      if backgrid4_4.status == STARTED:
        # check whether backgrid4_4 has been pressed
        if backgrid4_4.isClicked:
          if not backgrid4_4.wasClicked:
            backgrid4_4.timesOn.append(globalClock.getTime()) # store time of first click
            backgrid4_4.timesOff.append(globalClock.getTime()) # store time clicked until
          else:
            backgrid4_4.timesOff[-1] = globalClock.getTime() # update time clicked until
          None
          backgrid4_4.wasClicked = True  # if backgrid4_4 is still clicked next frame, it is not a new
click
        else:
          backgrid4_4.wasClicked = False  # if backgrid4_4 is clicked next frame, it is a new click
      else:
        backgrid4_4.wasClicked = False  # if backgrid4_4 is clicked next frame, it is a new click

      # *backgrid4_5* updates
      if backgrid4_5.status == NOT_STARTED and tThisFlip >= 1-frameTolerance:
        # keep track of start time/frame for later
        backgrid4_5.frameNStart = frameN  # exact frame index
        backgrid4_5.tStart = t  # local t and not account for scr refresh
        backgrid4_5.tStartRefresh = tThisFlipGlobal  # on global time
        win.timeOnFlip(backgrid4_5, 'tStartRefresh')  # time at next scr refresh
        backgrid4_5.setAutoDraw(True)
      if backgrid4_5.status == STARTED:
        # check whether backgrid4_5 has been pressed
        if backgrid4_5.isClicked:
          if not backgrid4_5.wasClicked:
            backgrid4_5.timesOn.append(globalClock.getTime()) # store time of first click
            backgrid4_5.timesOff.append(globalClock.getTime()) # store time clicked until
          else:
            backgrid4_5.timesOff[-1] = globalClock.getTime() # update time clicked until
          None
          backgrid4_5.wasClicked = True  # if backgrid4_5 is still clicked next frame, it is not a new
click
        else:
          backgrid4_5.wasClicked = False  # if backgrid4_5 is clicked next frame, it is a new click
      else:
        backgrid4_5.wasClicked = False  # if backgrid4_5 is clicked next frame, it is a new click
```

```
        # *backgrid4_6* updates
        if backgrid4_6.status == NOT_STARTED and tThisFlip >= 1-frameTolerance:
            # keep track of start time/frame for later
            backgrid4_6.frameNStart = frameN  # exact frame index
            backgrid4_6.tStart = t  # local t and not account for scr refresh
            backgrid4_6.tStartRefresh = tThisFlipGlobal  # on global time
            win.timeOnFlip(backgrid4_6, 'tStartRefresh')  # time at next scr refresh
            backgrid4_6.setAutoDraw(True)
        if backgrid4_6.status == STARTED:
            # check whether backgrid4_6 has been pressed
            if backgrid4_6.isClicked:
                if not backgrid4_6.wasClicked:
                    backgrid4_6.timesOn.append(globalClock.getTime()) # store time of first click
                    backgrid4_6.timesOff.append(globalClock.getTime()) # store time clicked until
                else:
                    backgrid4_6.timesOff[-1] = globalClock.getTime() # update time clicked until
                None
                backgrid4_6.wasClicked = True  # if backgrid4_6 is still clicked next frame, it is not a new
click
            else:
                backgrid4_6.wasClicked = False  # if backgrid4_6 is clicked next frame, it is a new click
        else:
            backgrid4_6.wasClicked = False  # if backgrid4_6 is clicked next frame, it is a new click

        # check for quit (typically the Esc key)
        if endExpNow or defaultKeyboard.getKeys(keyList=["escape"]):
            core.quit()

        # check if all components have finished
        if not continueRoutine:  # a component has requested a forced-end of Routine
            break
        continueRoutine = False  # will revert to True if at least one component still running
        for thisComponent in backComponents:
            if hasattr(thisComponent, "status") and thisComponent.status != FINISHED:
                continueRoutine = True
                break  # at least one component has not yet finished

        # refresh the screen
        if continueRoutine:  # don't flip if this routine is over or we'll get a blank screen
            win.flip()

    # -------Ending Routine "back"-------
    for thisComponent in backComponents:
        if hasattr(thisComponent, "setAutoDraw"):
            thisComponent.setAutoDraw(False)
    return_cue.stop()  # ensure sound has stopped at end of routine
    trials.addData('return_cue.started', return_cue.tStartRefresh)
    trials.addData('return_cue.stopped', return_cue.tStopRefresh)
    # the Routine "back" was not non-slip safe, so reset the non-slip timer
    routineTimer.reset()
    thisExp.nextEntry()
```

Τερματισμός Πειράματος

```
# completed 40.0 repeats of 'trials'
sys.stdout = original_stdout # Reset the standard output
```

```
# get names of stimulus parameters
if trials.trialList in ([], [None], None):
    params = []
else:
    params = trials.trialList[0].keys()
# save data for this loop
trials.saveAsExcel(filename + '.xlsx', sheetName='trials',
    stimOut=params,
    dataOut=['n','all_mean','all_std', 'all_raw'])
trials.saveAsText(filename + 'trials.csv', delim=',',
    stimOut=params,
    dataOut=['n','all_mean','all_std', 'all_raw'])

# Flip one final time so any remaining win.callOnFlip()
# and win.timeOnFlip() tasks get executed before quitting
win.flip()

# these shouldn't be strictly necessary (should auto-save)
thisExp.saveAsWideText(filename+'.csv', delim='semicolon')
thisExp.saveAsPickle(filename)
logging.flush()
# make sure everything is closed down
thisExp.abort()  # or data files will save again on exit
win.close()
core.quit()
```

13. Παράρτημα Β: Κώδικας MATLAB

Αρχικοποίηση

```
close all;
clear all;

% Start eeglab without GUI
eeglab nogui;
```

Φόρτωση και Μετασχηματισμός Δεδομένων

```
%% Load data
eeglab nogui;
eeg_demo = pop_biosig('pilot30.bdf');
exp_data = readtable('OpenBCI-RAW-2022-04-15_18-48-51_numbers.csv');

%% Get dataset information
eeg_channelnum = 16;
eeg_sample_rate = 125;
% Delete unneeded channels
exp_data=removevars(exp_data,{'AccelChannel0','AccelChannel1','AccelChannel2',...
    'Other','Other_1','Other_2','Other_3','Other_4','Other_5','Other_6',...
    'AnalogChannel0','AnalogChannel1','AnalogChannel2'});
```

```
%%
% Rename EEG channels
% Renames: T3 -> T7, T4 -> T8, T5 -> P7, T6 -> P8
eeg_channels = {'Fp1','Fp2','C3','C4','P7','P8','O1','O2','F7','F8','F3','F4','T7','T8','P3','P4'};
exp_data.Properties.VariableNames(2:17) = eeg_channels;

% Apply scale factor to EEG measurements
% Not used
eeg_gain = 24;
eeg_scale_factor = (4.5/eeg_gain)/(2^23-1); %(Volts/count);
%exp_data{:,2:17} = exp_data{:,2:17}*eeg_scale_factor;

% Create eeglab struct with data
eeg_exp = eeg_demo;
%struct_fields = fieldnames(eeg_exp);
%%
chanlogs=table2struct(chanlogs);
eeg_exp.chanlocs= chanlogs;

%%
% Edit fields
% Info on fields: https://sccn.ucsd.edu/~arno/eeglab/auto/eeg_checkset.html
eeg_exp.setname='eeg_desc';
eeg_exp.filename='filename_exp';
%eeg_exp.filepath=[dir_exp '\' filename_exp];
%eeg_exp.subject='';
%eeg_exp.group='';
%eeg_exp.condition='';
%eeg_exp.session=1;
%%eeg_exp.comments='';
eeg_exp.nbchan=eeg_channelnum;
%eeg_exp.trials=1;
%eeg_exp.pnts=size(exp_data,1);
eeg_exp.srate=eeg_sample_rate;
eeg_exp.xmin=0;
eeg_exp.xmax=857.9;
eeg_exp.times=transpose(str2double(exp_data{:,18}));
eeg_exp.data=str2double(exp_data{:,2:17});
eeg_exp.data=transpose(single(eeg_exp.data));
eeg_exp.data=eeg_exp.data*eeg_scale_factor;
eeg_exp.pnts=size(exp_data(:,1),1);
%%
channels_to_keep = [];
for i = 1:size(eeg_exp.chanlocs,1)
   if any(strcmp(eeg_channels,eeg_exp.chanlocs(i).labels))
      channels_to_keep(end+1) = i;
      %eeg_exp.chanlocs(i).labels
   end
end
eeg_exp.chanlocs = eeg_exp.chanlocs(channels_to_keep);

eeg_exp.ref='common';
eeg_exp.event = eeg_exp.event([]);
eeg_exp.urevent = eeg_exp.urevent([]);
%eeg_exp.eventdescription = '';
%epoch = [];
%epochdescription = [];
```

Εξαγωγή Μετρήσεων και Συμπερασμάτων

```
%%
eeg_exp = eeg_checkset(eeg_exp);  %check the dataset consistency
%%
eeg_exp = pop_reref( eeg_exp, []);  %average reference

%pop_eegplot(eeg_exp, 1, 0, 1);                    %plot the data
%%
eeg_exp = pop_resample(eeg_exp, 125);

%%
eeg_exp = pop_eegfiltnew(eeg_exp, 'locutoff',  1, 'hicutoff',  50);     %pass 1-50Hz

%%
eeg_exp.data(:,:) = detrend(eeg_exp.data(:,:)')';

%%
eeg_exp = pop_runica(eeg_exp, 'icatype', 'runica', 'extended',1,'interrupt','on');  %ICA
%%
pop_topoplot(eeg_exp, 0); % plot ICA

%%
eeg_exp=iclabel(eeg_exp)
round(eeg_exp.etc.ic_classification.ICLabel.classifications(:,:)*100)
eeg_exp.etc.ic_classification.ICLabel.classes
%%
eeg_exp = pop_subcomp(eeg_exp, 4, 1);   %(struct, component, 1)
%%
 EEGf = fft(eeg_exp.data);        %FFT
 banda = bandpass(EEGf, [8 13], 125);  %alpha band
 bandb = bandpass(EEGf, [14 26], 125);  %beta band
 bandg = bandpass(EEGf, [30 50], 125);  %gamma band
 bandd = bandpass(EEGf, [1 4], 125);    %delta band
 bandth = bandpass(EEGf, [4 8], 125);   %theta band
 powera = bandpower(banda);           %alpha band power
 powerb = bandpower(bandb);           %beta band power
 powerg = bandpower(bandg);           %gamma band power
 powerd = bandpower(bandd);           %delta band power
 powerth = bandpower(bandth);         %theta band power
 plot(powera);
 plot(powerb);
 plot(powerg);
 plot(powerd);
 plot(powerth);
%%
%t-test
%% The below includes alpha band only, the same has been run for all bands by replacing powera1
and powera2 variables
%from each population we use the mean value in each sample to determine if the mean value is
equal between the two populations
[h,p_welch,ci_diff,stats_w] = ttest2(powera1(:), powera2(:), 'Vartype','unequal');
%%
%Wilcoxon rank sum testcollapse
p_rs = ranksum(powera1(:), powera2(:));
x = powera1(:); y = powera2(:); n1 = numel(x); n2 = numel(y);
mu1 = mean(x); mu2 = mean(y); s1 = std(x,0); s2 = std(y,0);
sp = sqrt(((n1-1)*s1^2 + (n2-1)*s2^2)/(n1+n2-2));
d = (mu1 - mu2) / sp;
```

```
g = d * (1 - (3/(4*(n1+n2)-9)));
B = 5000; db = zeros(B,1);
for b = 1:B
   xb = x(randi(n1,[n1,1])); yb = y(randi(n2,[n2,1]));
   spb = sqrt(((n1-1)*var(xb,0) + (n2-1)*var(yb,0))/(n1+n2-2));
   db(b) = (mean(xb) - mean(yb)) / spb
end
ci_d = prctile(db,[2.5 97.5]);
```