

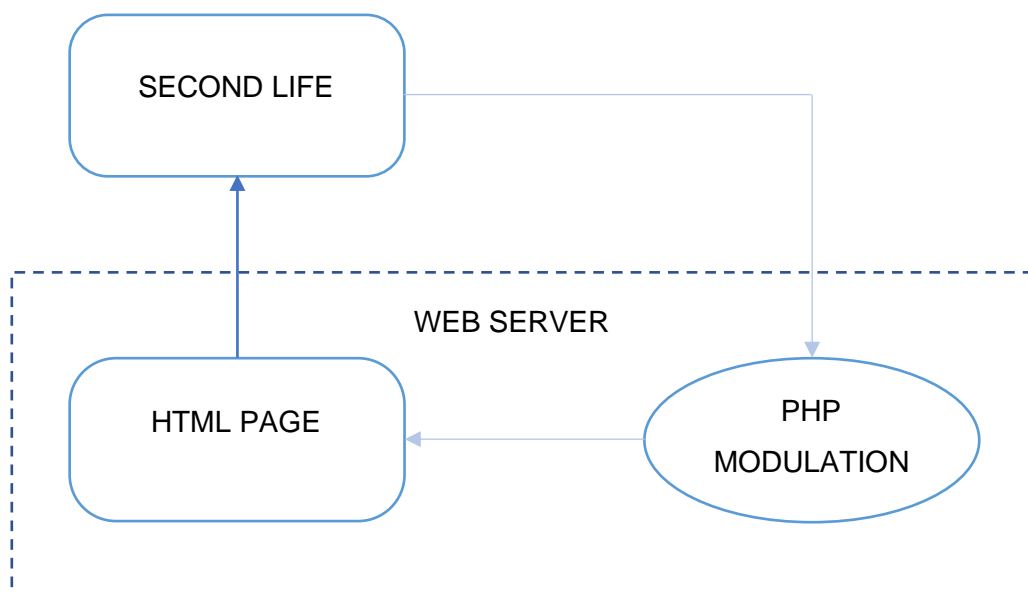
ΚΕΦΑΛΑΙΟ 4: Ανάλυση και Επεξεργασία Δεδομένων

4.1 ΕΙΣΑΓΩΓΗ

Το project στο σύνολο του τίθεται σε λειτουργία εντός του Second Life και ταυτόχρονα επικοινωνεί μέσω HTTP επικοινωνίας με έναν Web Server. Το παιχνίδι Simon Says, από τη στιγμή που ενεργοποιηθεί εντός του Second Life είναι ικανό να αναγνωρίσει και να διαχειριστεί τις αλληλεπιδράσεις του χρήστη είτε εντός του Second Life είτε μέσω του Web Browser. Ακόμη, έχει τη δυνατότητα να αναγνωρίζει εναλλαγές στον τρόπο που αλληλεπιδράει ο χρήστης με αυτό και να προσαρμόζεται κατάλληλα επιτρέποντας στον χρήστη να συνεχίσει το παιχνίδι του με εναλλαγή από Second Life σε Web Browser ή και αντίστροφα.

Για να καταστεί αυτό δυνατό, χρησιμοποιούνται αρχεία κώδικα (scripts) στο Second Life που εκτελούν το παιχνίδι Simon Says ενώ ταυτόχρονα επεξεργάζονται τις αλληλεπιδράσεις του χρήστη με αυτό αλλά και τις διαύλους επικοινωνίας που υπάρχουν.

Επίσης χρησιμοποιείται ένας Web Server (Host σε αυτήν την περίπτωση) όπου επικοινωνεί με το Second Life. Συγκεκριμένα, μια σελίδα HTML περιέχει το user interface και διαχειρίζεται τα δεδομένα που αποστέλλονται αλλά και που λαμβάνονται. Η HTML σελίδα αποστέλλει απευθείας τα δεδομένα στο Second Life ενώ λαμβάνει δεδομένα από αυτό μέσω μιας επεξεργασίας που περιλαμβάνει ένα αρχείο PHP, όπως φαίνεται και στο Γράφημα 4.1. Τόσο το Second Life όσο και ο Web Server είναι απαραίτητο να είναι συνδεδεμένοι σε μία σταθερή σύνδεση Internet ώστε να καταστεί δυνατή τόσο η εκτέλεση του παιχνιδιού Simon Says στο Second Life όσο και η επικοινωνία με τον Web Server.



Γράφημα 4.1. Η διάταξη επικοινωνίας του συνολικού Project.

4.2 SECOND LIFE

Στο Second Life η υλοποίηση και η λειτουργία του Project επιτυγχάνεται με τη χρήση 9 διαφορετικών αρχείων κώδικα (scripts) τοποθετημένα σε ισάριθμα αντικείμενα (objects) του Second Life. Αυτά τα αντικείμενα επιτελούν ξεχωριστή λειτουργία το κάθε ένα ενώ επικοινωνούν μεταξύ τους στην πλειοψηφία τους, ώστε να είναι δυνατή η διεκπεραίωση του παιχνιδιού.

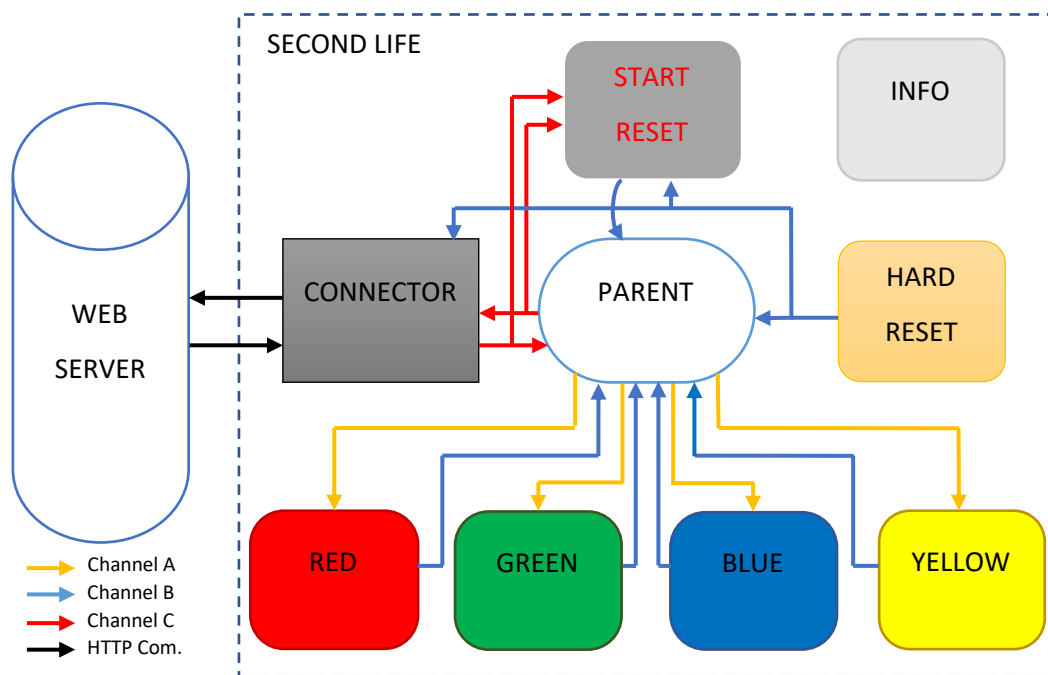
Σε ένα βασικό script στο Second Life υπάρχει ένα default state στο οποίο διεκπεραιώνονται τα ζητούμενα του κώδικα. Όμως ένα αντικείμενο μπορεί να έχει παραπάνω από μία καταστάσεις (states) στις οποίες εναλλάσσεται ανάλογα τα εξωτερικά ερεθίσματα ή προγραμματισμένες λειτουργίες. Κάθε state περιλαμβάνει ένα state_entry(){} το οποίο αποτελεί την αρχικοποίηση της συγκεκριμένης κατάστασης. Σε αυτήν γίνονται οι αρχικοποιήσεις των μεταβλητών που χρησιμοποιούνται στη συγκεκριμένη κατάσταση (state).

Η επικοινωνία μεταξύ των αντικειμένων γίνεται χάρη στην εντολή του Second Life lISay(channel,"text") ή lShout(channel,"text") με τις οποίες αποστέλλονται δεδομένα στο εκάστοτε channel και τη συνάρτηση listen με την οποία γίνεται παραλαβή δεδομένων.

Στο συγκεκριμένο Project γίνεται χρήση 3 διαφορετικών καναλιών επικοινωνίας (channels) μέσω των οποίων τα διαφορετικά αντικείμενα του Second Life μπορούν να επικοινωνήσουν. Αυτά τα 3 κανάλια φέρουν διαφορετική αρίθμηση το κάθε ένα, ενώ η αρίθμηση τους έχει επιλεχτεί τυχαία αλλά πάντα με γνώμονα την αποφυγή τυχαίας παρεμβολής. Αυτό συμβαίνει διότι μικροί αριθμοί ή συνεχόμενοι (πχ 1, 2, 3) μπορούν να χρησιμοποιούνται τυχαία από διαφορετικούς χρήστες στην περιοχή του Second Life που μπορεί να βρίσκεται ο χρήστης κι έτσι συμπτωματικώς να προκαλέσουν ανεπιθύμητες παρεμβολές στο σύστημα κι εν τέλει τη μη σωστή λειτουργία του.

Άρα για αυτόν το λόγο, επιλέχθηκαν ως κανάλια επικοινωνίας τα Channel A = -5243212, Channel B = -5243210 και Channel C = -5154789. Οι αρνητικές τιμές της αρίθμησης δεν προκαλούν κάποιο πρόβλημα ενώ με αυτόν τον τρόπο εξασφαλίζεται ακόμα περισσότερο η τυχαιότητα των αριθμών. Ο τρόπος επικοινωνίας στο Second Life φαίνεται στο Γράφημα 4.2.

Στο Γράφημα 4.2 γίνεται εμφανής η πολύπλοκη επικοινωνία που λαμβάνει χώρα μεταξύ των διαφορετικών αντικειμένων του Second Life. Ωστόσο, σε προσπάθεια να γίνει πιο απλή η φύση της επικοινωνίας μεταξύ των αντικειμένων, τα κανάλια επικοινωνίας ομαδοποιήθηκαν στις 3 προαναφερθείσες κατηγορίες. Η ομαδοποίηση έγινε με γνώμονα τις διαφορετικές λειτουργίες που λαμβάνουν χώρα στο Second Life και τη σχέση κάθε λειτουργίας με το κεντρικό αντικείμενο PARENT. Για να καταστεί δυνατή η επικοινωνία το αντικείμενο που επιθυμεί να αποστείλει δεδομένα χρησιμοποιεί την εντολή lISay(Channel, "text").



Γράφημα 4.2. Η επικοινωνία των αντικειμένων στο Second Life.

Προφανώς με αυτόν τον τρόπο ένα αντικείμενο μπορεί να αποστέλλει πολλαπλά μηνύματα σε ένα ή και σε πολλαπλά κανάλια επικοινωνίας ταυτόχρονα. Ακόμη, πολλαπλά αντικείμενα μπορούν να ακούσουν το ίδιο μήνυμα ταυτόχρονα και να δράσουν αναλόγως. Όμως, η απολαβή των δεδομένων αποτελεί πιο πολύπλοκο εγχείρημα καθώς χρησιμοποιείται συνάρτηση συμβάντος.

Συγκεκριμένα, στην `state_entry` του αντικειμένου που λαμβάνει δεδομένα, αρχικοποιείται μια μεταβλητή απολαβής (`handle`) για το συμβάν της ακρόασης (`listen event`) [1]. Στη συγκεκριμένη αρχικοποίηση, ορίζεται το κανάλι το οποίο το αντικείμενο απολαβής θα παρακολουθεί. Ένα αντικείμενο έχει τη δυνατότητα να παρακολουθεί παραπάνω από ένα κανάλια επικοινωνίας, άρα θα πρέπει να οριστούν περισσότερες μεταβλητές `listen_handle`.

Στη συνέχεια, εκτός του `state_entry`, δημιουργείται το event `listen()` εντός του οποίου το αντικείμενο μπορεί να διαχειριστεί τα διαφορετικά channel με τη χρήση μιας συνθήκης `if` αλλά και να διαχειριστεί διαφορετικά κείμενα που έχουν αποσταλεί στο συγκεκριμένο channel, πάλι με τη χρήση μιας εμφωλευμένης συνθήκης `if`, στην υπάρχουσα `if` του καθορισμού του channel. Άρα, ένα αντικείμενο μπορεί ταυτόχρονα να «ακούει» πολλαπλά κανάλια επικοινωνίας αλλά και να «ψάχνει» διαφορετικά μηνύματα μέσα σε αυτά τα κανάλια και ανάλογα τα μηνύματα τα οποία εντοπίζει, να δρα και αναλόγως.

4.2.1 ΛΕΙΤΟΥΡΓΙΑ ΕΠΙΚΟΙΝΩΝΙΑΣ ΠΑΙΧΝΙΔΙΟΥ SIMON SAYS

Η πρώτη λειτουργία είναι η επικοινωνία της κεντρικής πλακέτας PARENT με τα τέσσερα χρωματιστά κουμπιά και διεκπεραιώνεται με το κανάλι επικοινωνίας Channel A (-5243212). Μέσω αυτής της λειτουργίας το Second Life είναι δυνατόν να αποκτήσει υπόσταση και να μπορεί να παράγει τυχαίες χρωματικές ακολουθίες αλλά και να τις αναπαράγει στον χρήστη εντός του Second Life.

Το αντικείμενο PARENT επικοινωνεί με τα 4 χρωματιστά αντικείμενα RED, GREEN, BLUE και YELLOW ονοματίζοντας στον δίαυλο επικοινωνίας το όνομα εκείνου του αντικειμένου που επιλέχθηκε και πρέπει να ανάψει. Για παράδειγμα, στο κανάλι A μπορεί να εμφανιστεί το μήνυμα «B» το οποίο υποδηλώνει το χρώμα μπλε (BLUE). Στη συγκεκριμένη περίπτωση επικοινωνίας το όνομα ενός χρώματος της ακολουθίας αναφέρεται μόνο με το πρώτο γράμμα του χρώματος στα Αγγλικά. Στην πραγματικότητα, ολόκληρη η ακολουθία των χρωμάτων που παράγονται τυχαία στο αντικείμενο PARENT αναφέρεται μόνο με τα πρώτα γράμματα των χρωμάτων. Δηλαδή μια παραγόμενη ακολουθία θα μπορούσε να είναι η «RRYBR» η οποία αντιστοιχεί στα χρώματα RED, RED, YELLOW, BLUE, RED. Αυτός ο τρόπος απεικόνισης επιλέχτηκε διότι αυτού του είδους η απεικόνιση της χρωματικής ακολουθίας είναι πιο εύκολα επεξεργάσιμος κυρίως στον WEB SERVER όπου και αποστέλλεται αργότερα και προφανώς καταλαμβάνει μικρότερη μνήμη.

Η αποστολή του κάθε χαρακτήρα του χρώματος συμβαίνει ανά 1 δευτερόλεπτο ενώ υπεύθυνη για αυτή τη λειτουργία είναι η συνάρτηση pattern_SHOUT. Σε αυτήν, παράγεται κάθε φορά το επόμενο χρώμα της χρωματικής ακολουθίας. Η επιλογή του κάθε χρώματος γίνεται τυχαία. Τα τέσσερα διαθέσιμα χρώματα υπάρχουν σε μορφή λίστας [2] και μια εντολή lIfRand επιλέγει έναν τυχαίο ακέραιο αριθμό από το 0 έως το 3. Ο κάθε αριθμός υποδηλώνει και μια θέση στη συγκεκριμένη λίστα, πρακτικά δηλαδή ένα χρώμα. Η εντολή lIfRand επιλέγει τον αριθμό μέσω ενός προκαθορισμένου εύρους αριθμών ενώ η ακολουθία των τυχαίων αριθμών είναι σε κοινή χρήση σε ολόκληρη τη διαδικασία προσομοίωσης του Second Life [3]. Άρα η κάθε επιλογή είναι ψευδοτυχαία αφού γίνεται από προκαθορισμένη δεξαμενή αριθμών.

Η καθυστέρηση στην αποστολή των χαρακτήρων της ακολουθίας συμβαίνει διότι ένα μόνο χρώμα της πλακέτας στο Second Life θα πρέπει να είναι ανοιχτό τη φορά. Αυτή η καθυστέρηση του ενός δευτερολέπτου προκύπτει από την εντολή lISleep(1.0)[4] που ορίζει το χρόνο για τον οποίο ο εκτελέσιμος κώδικας θα παραμείνει ανενεργός, μέχρι να ξαναξεκινήσει από το σημείο που σταμάτησε. Εάν ολόκληρη η χρωματική ακολουθία στελνόταν ταυτόχρονα σε κάθε κουμπί τότε θα άναβαν όλα μαζί μη κάνοντας δυνατή την επιτυχή προβολή της χρωματικής ακολουθίας.

Στα χρωματικά κουμπιά ωστόσο, αφού έχει διασφαλιστεί η σωστή αναπαραγωγή της χρωματικής ακολουθίας από το αντικείμενο Parent, δεν απαιτείται σε αυτά περαιτέρω επεξεργασίας της ακολουθίας, παρά μόνο την προβολή τους. Άρα, το κάθε ένα αντικείμενο, μόλις λάβει στο κανάλι επικοινωνίας, κείμενο με το όνομα του, αλλάζει χρώμα για χρονικό διάστημα 0.5 δευτερολέπτου και επανέρχεται στην αρχική του κατάσταση. Το χρονικό όριο του μισού δευτερολέπτου σε αυτήν την περίπτωση επιτυγχάνεται μέσω της ξεχωριστής κλήσης της συνάρτησης timer() που εκτελείται μέσω της εντολής ISetTimerEvent(toff_unit). Η τιμή της μεταβλητής toff_unit ορίζει το χρονικό διάστημα της καθυστέρησης. Εδώ, εφόσον αλλάζει το χρώμα ολόκληρου του αντικειμένου που υποδεικνύει το χρώμα της ακολουθίας, αλλάζει και η ορισμένη κατάσταση. Έτσι από default state, τα χρωματιστά κουμπιά RED, GREEN, BLUE και YELLOW μεταβαίνουν στα αντίστοιχα state RED_BUTTON, state GREEN_BUTTON, state BLUE_BUTTON και state YELLOW_BUTTON. Σε αυτές τις καταστάσεις το μόνο που αλλάζει είναι το χρώμα των κουμπιών όπου από τη σκοτεινή (DARK) εκδοχή τους, μεταβαίνουν στη φωτεινή (BRIGHT).

Ωστόσο, αξίζει να σημειωθεί πως όση ώρα το κάθε κουμπί βρίσκεται στη διαφορετική κατάσταση (state) εκτός της προκαθορισμένης (default) δε δύναται να εκτελέσει άλλες εντολές που μπορεί να ζητήσει ο χρήστης ταυτόχρονα. Με αυτόν τον τρόπο εξασφαλίζεται μια προστασία απέναντι σε προβληματική χρήση ορισμένων χρηστών που ίσως επιθυμούν να επιλέγουν πολλαπλά κουμπιά ταυτόχρονα.

Επίσης, σε αυτήν τη λειτουργία επικοινωνίας του παιχνιδιού Simon Says εμπίπτει μόνο η μονόδρομη επικοινωνία του κεντρικού προγράμματος PARENT με τα τέσσερα χρωματιστά κουμπιά επιλογών. Η αντίθετη διαδρομή, από τα τέσσερα κουμπιά προς το αντικείμενο PARENT καλύπτεται από την επόμενη λειτουργία επικοινωνίας. Αυτό συμβαίνει επειδή η αρχική επικοινωνία του PARENT προς τα κουμπιά εξυπηρετεί τη βασική αρχή ύπαρξης του παιχνιδιού.

Η συγκεκριμένη λειτουργία θεωρείται η πιο σημαντική λειτουργία καθώς χωρίς αυτήν δεν υφίσταται Second Life. Έτσι, επιλέχθηκε αυτή η βασική λειτουργία να έχει δικό της ξεχωριστό κανάλι επικοινωνίας. Παρόλα αυτά, όλες οι επικοινωνίες, δυνητικά θα μπορούσαν να εξυπηρετούνται από μόνο ένα κανάλι επικοινωνίας και να μην επηρεάζεται με κάποιον τρόπο η εύρυθμη λειτουργία του project. Όμως με αυτόν τον τρόπο ο κώδικας γίνεται πιο πολύπλοκος και το κυριότερο, πιο δύσκολος στην ανάγνωση και μελλοντική επεξεργασία.

4.2.2 ΛΕΙΤΟΥΡΓΙΑ ΔΕΥΤΕΡΕΥΟΝΤΩΝ ΕΠΙΚΟΙΝΩΝΙΩΝ

Η δεύτερη λειτουργία περιλαμβάνει την επικοινωνία σχεδόν όλων των κουμπιών με το κεντρικό αντικείμενο PARENT. Σε αυτήν τη λειτουργία τα 4 χρωματιστά κουμπιά, το κουμπί HARD RESET, το κουμπί START/RESET και το κουμπί CONNECTOR επικοινωνούν με το κεντρικό αντικείμενο PARENT αλλά και μεταξύ τους, χρησιμοποιώντας το κανάλι επικοινωνίας Channel B (-5243210).

Σε αυτή τη λειτουργία μεταδίδονται μηνύματα που παράγονται από την αλληλεπίδραση του χρήστη με το «περιβάλλον χρήστη». Αυτά τα μηνύματα περιλαμβάνουν μηνύματα SOFT RESET και HARD RESET, το μήνυμα START καθώς και τα μηνύματα επιλογής (touch) των τεσσάρων χρωματιστών κουμπιών.

Αρχικά, τα τέσσερα χρωματιστά κουμπιά RED, GREEN, BLUE και YELLOW πέρα από την προβολή της χρωματικής ακολουθίας, έχουν τη δυνατότητα να δεχτούν τα δεδομένα εισόδου του χρήστη μέσω της επιλογής Touch του χρήστη σε κάθε κουμπί. Έτσι, με το πάτημα κάθε κουμπιού από τον χρήστη, το κουμπί αποστέλλει στο κεντρικό αντικείμενο PARENT το όνομα του. Άρα, άμα πατηθεί το κουμπί GREEN θα αποσταλεί σα μήνυμα στο κανάλι B το κείμενο «GREEN». Σημειώνεται ότι δε θα αποσταλεί μόνο το αρχικό γράμμα «G» του χρώματος καθώς η επεξεργασία της συγκεκριμένης εντολής γίνεται από το κεντρικό αντικείμενο PARENT το οποίο διατηρεί αποθηκευμένη τη λίστα των διαθέσιμων χρωμάτων, και όχι στον WEB SERVER ο οποίος δεν έχει δυνατότητα επεξεργασίας των δεδομένων με αποστολή ενός χρώματος τη φορά. Επίσης πρέπει να τονιστεί πως με το πάτημα ενός από τα συγκεκριμένα χρωματιστά κουμπιά, το εκάστοτε κουμπί δεν μεταβαίνει στην ειδική κατάσταση (state) στην οποία αναβοσβήνει. Επιλέχθηκε αυτή η προσέγγιση καθώς παρατηρήθηκε επιπλέον καθυστέρηση με τη χρήση της συγκεκριμένης κατάστασης. Η συγκεκριμένη παρατήρηση δεν επηρεάζει τις υπόλοιπες περιπτώσεις όπου γίνεται κλήση της κατάστασης, καθώς σε εκείνες τις περιπτώσεις δεν υπάρχει άμεση αλληλεπίδραση του χρήστη με το αντικείμενο στο Second Life. Άρα, όταν ο χρήστης χρησιμοποιεί το παιχνίδι μόνο εντός Second Life, τα γραφικά και τα animation των διαφόρων κουμπιών πρέπει να είναι όσο το δυνατόν πιο γρήγορα αποκριτικά ή και να απουσιάζουν τελείως, όπως στη συγκεκριμένη περίπτωση. Στις περιπτώσεις που το παιχνίδι δε διαδραματίζεται εξ ολοκλήρου στο Second Life, άσχετα αν η εκτέλεση του λαμβάνει χώρα πάντα εντός αυτού, τα γραφικά και τα animation δεν επηρεάζουν κάπου την εμπειρία παιχνιδιού κι έτσι συμπεριλήφθηκαν ή δε δέχτηκαν ειδική μεταχείριση εν αντιθέσει με πριν.

Επίσης, επικοινωνία με το κεντρικό αντικείμενο PARENT μέσω του καναλιού channel B πραγματοποιεί το κουμπί START/RESET. Το συγκεκριμένο κουμπί διεκπεραιώνει δύο διαφορετικές λειτουργίες ενώ και το ίδιο το κουμπί κατέχει δύο διαφορετικές εμφανίσεις και φύσεις. Το κουμπί εναλλάσσει λειτουργία και εμφάνιση ανάλογα με το αν έχει ξεκινήσει ήδη το παιχνίδι ή αν έχει τερματιστεί.

Συγκεκριμένα, αυτή η λογική συνθήκη ανάγεται στο εάν έχει πατήσει ήδη μια φορά το κουμπί ο χρήστης ή όχι. Επομένως, εάν προβάλλεται το κουμπί START τότε σημαίνει ότι δεν έχει πατηθεί το κουμπί από τον χρήστη, τουλάχιστον όχι πριν την ολοκλήρωση του προηγούμενου παιχνιδιού. Με το πάτημα του κουμπιού τώρα, η όψη του κουμπιού αλλάζει και εμφανίζεται αυτή του RESET. Έτσι, αν πατηθεί ακόμα μια φορά το κουμπί, η όψη του επανέρχεται στην αρχική όψη αυτή του START. Η εναλλαγή των όψεων και της λειτουργίας ελέγχεται χάρη στη μεταβλητή `stateofSTART` η οποία λαμβάνει την τιμή 0 όταν είναι σε κατάσταση ηρεμίας και προβάλλει τη λέξη START ενώ γίνεται 1 όταν πατηθεί το κουμπί START. Με την επαναφορά του προγράμματος στην αρχική του κατάσταση, αυτή η μεταβλητή επανααρχικοποιείται και αποκτάει ξανά την τιμή 0.

Η όψη του κουμπιού START με το που πατηθεί, πέρα από τη μετάλλαξη που υπόκειται σε κουμπί RESET, στέλνει το μήνυμα «START» στο κεντρικό αντικείμενο PARENT. Εκεί, μέσω μιας δομής ελέγχου IF επιτρέπεται να ξεκινήσει η παραγωγή της τυχαίας ακολουθίας χρωμάτων. Εάν δεν έχει πατηθεί το κουμπί START τόσο η χρωματική ακολουθία όσο και τα 4 χρωματιστά κουμπιά δε μπορούν να λειτουργήσουν.

Η εναλλακτική όψη η οποία εμφανίζει τη λέξη RESET εμφανίζεται με το που πατηθεί προηγουμένως το κουμπί START. Έτσι, το κουμπί RESET είναι διαθέσιμο καθ' όλη τη διάρκεια παιχνιδιού ενώ επιστρέφει στην αρχική του κατάσταση με το που πατηθεί ή όταν ο χρήστης χάσει στο παιχνίδι. Όταν πατηθεί το κουμπί RESET τότε μεταδίδει στο κανάλι Channel B το μήνυμα «RESET» το οποίο δέχεται μόνο το κεντρικό αντικείμενο PARENT. Αυτό έχει σαν αποτέλεσμα το κεντρικό αντικείμενο PARENT να επαναφέρεται (Reset). Ταυτόχρονα επαναφέρεται και το κουμπί START/RESET, προβάλλοντας τη λέξη START ξανά. Αυτή η διαδικασία φέρει την ονομασία SOFT RESET ώστε να ξεχωρίζει από τη διαδικασία HARD RESET που επιφέρει επιπλέον επιπτώσεις και σε άλλα αντικείμενα.

Το HARD RESET κουμπί οπότε, είναι το τελευταίο κουμπί που κάνει χρήση του καναλιού επικοινωνίας Channel B. Μέσω αυτού, όταν πατηθεί, αποστέλλει το μήνυμα «HARD_RESET» σε πολλά αντικείμενα του project στο Second Life. Αυτά περιλαμβάνουν τα αντικείμενα PARENT, CONNECTOR και START/RESET. Στο κεντρικό αντικείμενο PARENT έχει ίδια χρήση με το προηγούμενο SOFT RESET, δηλαδή επαναφέρει τον κώδικα του συγκεκριμένου αντικειμένου. Αυτή η επαναφορά σημαίνει εμμέσως τη διακοπή του παιχνιδιού και την αρχικοποίησή του, οπότε επανέρχεται και ο κώδικας στο κουμπί START/RESET με το μήνυμα του HARD RESET. Τέλος, το συγκεκριμένο μήνυμα γίνεται αντιληπτό και από το κουμπί CONNECTOR το οποίο με αυτόν τον τρόπο αποδεσμεύει URL που είχε ήδη παράγει για την HTTP επικοινωνία.

Με αυτόν τον τρόπο δε γίνεται επαναφορά του script του κουμπιού CONNECTOR καθώς δεν υπάρχει ανάγκη για αυτό. Ακόμη, το κουμπί CONNECTOR θα αναπαράγει καινούργιο URL αν πατηθεί αυτό το

κουμπί. Σε πιθανή δεύτερη συνεχόμενη επιλογή του κουμπιού CONNECTOR χωρίς να έχει μεσολαβήσει HARD RESET τότε απλά προβάλλεται το προηγούμενος παραχθέν URL και δεν παράγεται καινούργιο με γνώμονα την οικονομία πόρων. Η αποδέσμευση των URL δεν αποτελεί αμελητέο σκοπό καθώς τα URL καταλαμβάνουν φυσική υπόσταση στα region του Second Life και καταναλώνουν ψηφιακούς πόρους οι οποίοι δεν είναι ανεξάντλητοι αλλά σχετικά πολύτιμοι.

4.2.3 ΛΕΙΤΟΥΡΓΙΑ ΜΕΤΑΦΟΡΑΣ ΔΕΔΟΜΕΝΩΝ ΑΠΟ ΚΑΙ ΠΡΟΣ ΤΗΝ HTTP ΣΥΝΔΕΣΗ

Η τελευταία μορφή λειτουργία επικοινωνίας η οποία διεκπεραιώνεται εντός του Second Life περιλαμβάνει τη διαχείριση και μεταφορά των δεδομένων που είτε προέρχονται από τον Web Server είτε αποστέλλονται σε αυτόν. Ωστόσο αυτή η μορφή επικοινωνίας σε καμία περίπτωση δεν καλύπτει την ολική HTTP επικοινωνία με τον Web Server αλλά θα πρέπει να θεωρείται ως τη διαδικασία επεξεργασίας των δεδομένων πριν αποσταλούν στο διαδίκτυο ή αφού έχουν καταφτάσει από εκεί. Το κανάλι επικοινωνίας που χρησιμοποιείται είναι το Channel C (-5154789) και περιλαμβάνει την επικοινωνία μεταξύ των αντικειμένων PARENT, CONNECTOR και START/RESET. Προφανώς και δεδομένα που καταφτάνουν στο κεντρικό αντικείμενο PARENT επηρεάζουν και τα αντικείμενα RED, GREEN, BLUE και YELLOW. Όμως, το project έχει σχεδιαστεί με τέτοιο τρόπο ώστε οι λειτουργίες προβολής ενός χρώματος σε κάθε κουμπί να είναι ανεξάρτητες με τον τρόπο παιχνιδιού και να δουλεύουν με τον ίδιο τρόπο από όπου κι αν αλληλεπιδράει ο χρήστης.

Επομένως, κατά τη μεταφορά δεδομένων από τον Web Server προς τα αντικείμενα του Second Life, το αντικείμενο CONNECTOR μέσω του παραχθέντος URL, εντοπίζει τα παραγόμενα headers που δημιουργούνται στην HTML σελίδα και μέσω αυτού κατηγοριοποιεί τα μηνύματα σύμφωνα με τα 4 χρώματα «RED», «GREEN», «BLUE» και «YELLOW» αλλά και του μηνύματος «START».

Στο αντικείμενο START/RESET, το μήνυμα «START» προκαλεί την εναλλαγή της όψης του αντικειμένου στην όψη RESET. Παρόλα αυτά δεν προκαλεί νέα αποστολή του μηνύματος στο αντικείμενο PARENT καθώς αυτό θα προκαλούσε επιπλέον καθυστερήσεις αφού το μήνυμα θα έπρεπε να ληφθεί από το START/RESET και μετά να ξανααποσταλεί στο κεντρικό αντικείμενο του PARENT. Για αυτόν τον λόγο το μήνυμα «START» αποστέλλεται από τον CONNECTOR τόσο στο START/RESET όσο και στο αντικείμενο PARENT.

Στο αντικείμενο PARENT εάν φτάσει ένα μήνυμα χρώματος, προφανώς εφόσον έχει μεσολαβήσει πάτημα του κουμπιού START είτε στην HTML σελίδα είτε εντός του Second Life, αυτό αποστέλλει την πληροφορία σε ένα από τα 4 χρωματιστά κουμπιά επιλογής. Σε αυτήν την περίπτωση δεν υπάρχει λόγος απευθείας αποστολής της πληροφορίας στα κουμπιά επιλογής αφού έτσι κι αλλιώς θα υπάρξει

επεξεργασία των δεδομένων στο αντικείμενο PARENT. Έτσι, ο έλεγχος ταυτοποίησης των δεδομένων που εισέρχονται από την HTTP επικοινωνία σε σχέση με την ήδη παραχθείσα χρωματική ακολουθία, γίνεται στο κεντρικό αντικείμενο PARENT.

Από το συγκεκριμένο αντικείμενο, όμως, πάλι, ξεκινάει και η ροή δεδομένων προς την αντίθετη κατεύθυνση με προορισμό τον Web Server. Εκεί, δημιουργούνται δύο κατηγορίες δεδομένων που αποστέλλονται. Η πρώτη αφορά τα δεδομένα της χρωματικής ακολουθίας που έχει παραχθεί, τα οποία έχουν τη μορφή «RRYRBG» με το κάθε γράμμα να υποδηλώνει ένα από τα τέσσερα διαθέσιμα χρώματα. Έτσι, αυτή η ακολουθία αποστέλλεται με τη μορφή μηνύματος στο αντικείμενο CONNECTOR από το οποίο διαβιβάζεται στον Web Server.

Εδώ θα πρέπει να αναφερθεί πως ο κώδικας του αντικειμένου PARENT, χρησιμοποιώντας τη μεταβλητή `waitingForInput` ορίζει την κατάσταση στην οποία βρίσκεται το παιχνίδι και ως αποτέλεσμα τη συνάρτηση που θα εκτελεσθεί στη συνέχεια. Όταν η μεταβλητή `waitingForInput` έχει τιμή `FALSE` τότε το αντικείμενο εκτελεί τη συνάρτηση `pattern_SHOUT` στην οποία παράγεται η χρωματική ακολουθία. Εάν η μεταβλητή `waitingForInput` έχει τιμή `TRUE` τότε σημαίνει ότι έχει ήδη παράγει μία χρωματική ακολουθία και τώρα βρίσκεται σε αναμονή δεδομένων εισόδου από τον χρήστη, είτε έρθουν αυτά μέσω Web Server είτε εντός του `Second Life`. Έτσι, με την πρώτη εμφάνιση των δεδομένων εισόδου, εκτελείται η συνάρτηση `pattern_INPUT` κατά την οποία τα δεδομένα εισόδου συγκρίνονται με την ακολουθία χρωμάτων και ελέγχεται κατά πόσο είναι ίδιες. Αν δεν είναι ίδιες το παιχνίδι τερματίζεται. Στην συγκεκριμένη συνάρτηση όμως λαμβάνει χώρα και ο υπολογισμός του σκορ του χρήστη. Υπενθυμίζεται ότι μόνο ο υπολογισμός του σκορ συμβαίνει και στο `Second Life` και στον Web Server.

Η άλλη κατηγορία αφορά την περίπτωση που ο χρήστης, χρησιμοποιώντας το `Second Life` μέσω του Web Server, κάνει λάθος και χάσει στο παιχνίδι. Τότε, η ένδειξη ότι υπήρξε λάθος προκύπτει στο αντικείμενο PARENT με τη μορφή του χαρακτήρα «Ο» υποδηλώνοντας το `GAME OVER`. Αυτό το μήνυμα θα αποσταλεί πάλι στον CONNECTOR ο οποίος με τη σειρά του θα το μεταβιβάσει στον Web Server. Τέλος, ο χαρακτήρας «Ο» αποστέλλεται και στο κουμπί `START/RESET` υποδεικνύοντας του το τέλος του παιχνιδιού και άρα την επαναφορά του στην αρχική του κατάσταση και στην όψη `START`.

4.2.4 TO ANTIKEIMENO INFO

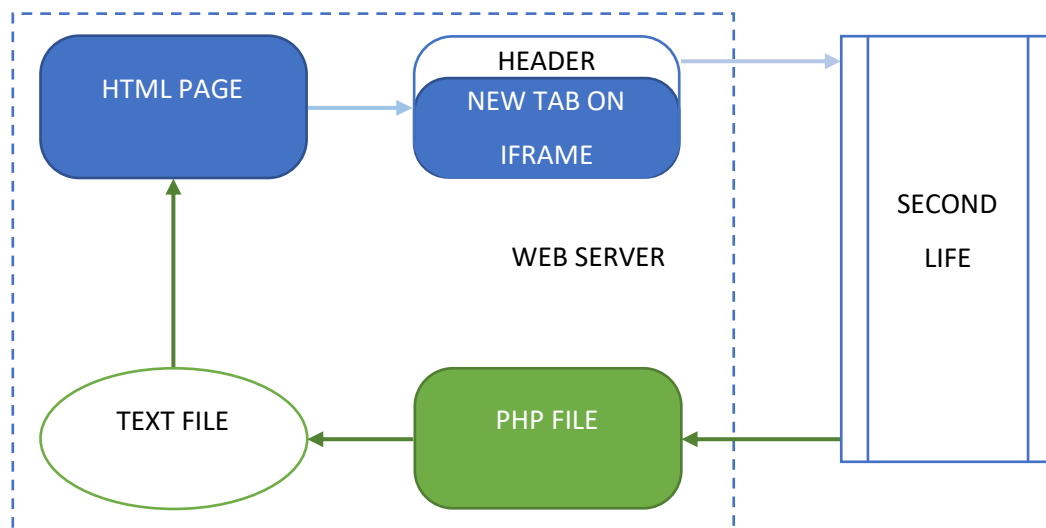
Το κουμπί-αντικείμενο `INFO` αποτελεί μια ξεχωριστή κατηγορία από μόνο του καθώς δεν επικοινωνεί με κάποιον τρόπο με τα υπόλοιπα αντικείμενα του `Second Life` αλλά και ούτε με τον Web Server. Παρόλα αυτά, ο τρόπος ο οποίος εμφανίζει τα παράθυρα με πληροφορίες για τον χρήστη αποτελούν

μια μορφή επικοινωνίας ελαφρώς πιο πολύπλοκη από τα βασικά μηνύματα χρήστη στο μηδενικό κανάλι (παράθυρο επικοινωνίας, chat) που δημιουργούν τα υπόλοιπα κουμπιά. Αυτό συμβαίνει διότι το συγκεκριμένο κουμπί χρησιμοποιεί την εντολή `Dialog()` η οποία εμφανίζει τα παράθυρα με τις πληροφορίες αλλά προβάλλει ταυτόχρονα και μια λίστα από κουμπιά. Ωστόσο, αυτά τα παράθυρα εμφανίζονται στον χρήστη μέσω επικοινωνίας στο κανάλι 1000 το οποίο ορίζεται εντός του script. Άρα, υπάρχει μεταφορά πληροφορίας στον χρήστη μέσω του καναλιού 1000 το οποίο δεν εμφανίζεται στο κεντρικό παράθυρο του chat (με αριθμό καναλιού 0) αλλά σε ένα τεχνητό παράθυρο. Όμως, όλες οι αλληλεπιδράσεις εντός αυτού του νέου παραθύρου μεταφέρονται στο κανάλι 1000 το οποίο όμως κανένα άλλο αντικείμενο δεν το χρησιμοποιεί στο Project.

4.3 WEB SERVER

Στον Web Server εξειδικευμένα αρχεία HTML και PHP έχουν την δυνατότητα να παράγουν δεδομένα για το Second Life αλλά και να μπορούν να δέχονται και να αναπαράγουν δεδομένα από το Second Life.

Συγκεκριμένα, μια HTML σελίδα αποτελεί το κέντρο διαχείρισης των δεδομένων είτε αυτά προέρχονται από το Second Life είτε αποστέλλονται εκεί. Η HTML σελίδα χρησιμοποιεί ταυτόχρονα μία σελίδα PHP ενώ κατέχει εμφωλευμένα αρχεία κώδικα CSS και Javascript. Η διάρθρωση του Web Server φαίνεται στο Γράφημα 4.3.



Γράφημα 4.3. Απεικόνιση της επικοινωνίας εντός του Web Server.

Υπενθυμίζεται ότι τα αρχεία που χρησιμοποιούνται στον Διαδικτυακό Server, βρίσκονται σε δωρεάν Host της υπηρεσίας «<https://www.000webhost.com>» η οποία προσφέρει δωρεάν θέση σε Server.

4.3.1 ΔΕΔΟΜΕΝΑ ΠΡΟΣ SECOND LIFE

Για να επιτευχθεί η αποστολή δεδομένων προς το Second Life γίνεται χρήση αποκλειστικά της HTML σελίδας και όχι του PHP αρχείου τα οποία βρίσκονται στον Web Server. Προφανώς για να μπορέσει να υπάρξει επικοινωνία με το Second Life, απαιτείται σταθερή σύνδεση στο διαδίκτυο, τόσο του Second Life όσο και του Web Server. Ακόμη απαιτείται το παιχνίδι Simon Says να είναι ενεργό και να εκτελείται στο Second Life ενώ επίσης θα πρέπει να υπάρχει διαθέσιμο URL το οποίο θα χρησιμοποιηθεί για την HTTP επικοινωνία.

4.3.1.1 ΠΑΡΑΓΩΓΗ ΣΤΗ ΣΕΛΙΔΑ HTML

Ο χρήστης, όπως προαναφέρθηκε, έχει τη δυνατότητα να παίξει το παιχνίδι Simon Says μέσω ενός Web Browser. Το User Interface της σελίδας HTML που χρησιμοποιείται, προσφέρει τα 4 χρωματιστά κουμπιά που αποτελούν και τα κουμπιά επιλογής, καθώς και ένα κουμπί «START GAME». Κάθε φορά που ο χρήστης πατάει ένα από αυτά τα 5 κουμπιά, εντός της σελίδας HTML εκτελείται μια συνάρτηση JavaScript η οποία είναι εμφωλευμένη στον κώδικα HTML. Στην πραγματικότητα, τα 4 χρωματιστά κουμπιά εκτελούν τη συνάρτηση SimonGame() με το συμβάν onclick, όπως φαίνεται στην Εικόνα 4.1, ενώ το κουμπί «START GAME» εκτελεί τη συνάρτηση StartGame(), όπως φαίνεται στην Εικόνα 4.2.

```
<div id="PRESSR"class=REDButton onclick="SimonGame('RED')"></div>  
<div id="PRESSG"class=GREENButton onclick="SimonGame('GREEN')"></div>
```

Εικόνα 4.1. Εκτέλεση της συνάρτησης SimonGame με το κλικάρισμα του κόκκινου ή του πράσινου κουμπιού.

```
<button id="button1"class="button1" onclick="StartGame()">START GAME</button>
```

Εικόνα 4.2. Εκτέλεση της συνάρτησης StartGame με το κλικάρισμα του κουμπιού "START GAME".

Στην Εικόνα 4.1 γίνεται εμφανές ότι η εκτέλεση της συνάρτησης SimonGame() με το πάτημα του κόκκινου κουμπιού (Το id «PRESSR» αντιστοιχεί στο κόκκινο κουμπί, το «PRESSG» στο πράσινο, το «PRESSB» στο μπλε και τέλος το «PRESSY» στο κίτρινο), γίνεται με δεδομένο εισόδου της συνάρτησης, το κείμενο «RED». Αντιθέτως, η συνάρτηση StartGame() εκτελείται χωρίς δεδομένα εισόδου. Αυτό συμβαίνει διότι η συνάρτηση SimonGame() είναι κοινή και για τα τέσσερα έγχρωμα κουμπιά και λειτουργεί με τον ίδιο τρόπο και για τα τέσσερα. Η συνάρτηση StartGame() όμως λειτουργεί αποκλειστικά με το κουμπί «START GAME», οπότε δεν απαιτούνται δεδομένα εισόδου που να ξεκαθαρίζουν ποιο κουμπί πατήθηκε.

4.3.1.2 Η ΕΝΤΟΛΗ WINDOW.OPEN

Ωστόσο, ακόμη μια ειδοποιός διαφορά μεταξύ των δύο συναρτήσεων, είναι ότι η συνάρτηση StartGame και η συνάρτηση SimonGame έχουν διαφορετικές χρήσεις ενώ μόνο ένα κομμάτι και των δύο συναρτήσεων είναι κοινό κι αυτό είναι η χρησιμοποίηση της εντολής window.open όπως φαίνεται και από τις εικόνες Εικόνα 4.3 και Εικόνα 4.4.

```
function SimonGame(str){
    step++;
    side_step = step;
    window.open(slurl + "?" + str, "win1");
    side_score = DataLength;
    loadDoc();
    if(step > score){
        score++;
    }
    if (step == DataLength){
        document.getElementById("button2").innerHTML = step;
        step=0;
    }
}
```

Εικόνα 4.3. Η συνάρτηση SimonGame

```
function StartGame(){
    window.open(slurl + "?" + "START", "win1");//We send "RED" because it doesn't matter what color is sent. SL code begins
    // with every color as an input data
    //-----Animation for the START GAME button(button1) and SCORE button(button2)-----//
    document.getElementById('button1').style.transform = 'translateY(-50%)';
    document.getElementById('button1').style.visibility = 'hidden';
    setTimeout(function() {document.getElementById('button2').style.transform = 'translateY(40%)';}, 300);
    setTimeout(function() {document.getElementById('button2').style.visibility = 'visible';}, 300);
    document.getElementById("button2").innerHTML = "";
    setTimeout(function() {loadDoc();},2000);// Delay of 2s just to be sure we have got the php data sent to the html page
}
```

Εικόνα 4.4. Η συνάρτηση StartGame

Η εντολή window.open() δημιουργεί καινούργιο παράθυρο στον Web Browser που χρησιμοποιείται για την αποστολή δεδομένων μέσω HTTP Request, ενώ στο εσωτερικό της παρένθεσης ορίζεται η διεύθυνση που θα φέρει αυτό το νέο παράθυρο. Στην περίπτωση του συγκεκριμένου project, χρειάζεται να ανοίξει ένα νέο παράθυρο ονόματος (id) «win1» το οποίο θα φέρει σε διεύθυνση μια σελίδα αποτελούμενη από τρία μέρη. Το πρώτο μέρος αυτής της διεύθυνσης θα είναι το URL που έχει παράγει το Second Life. Το συγκεκριμένο μέρος αποτελεί το περιεχόμενο της μεταβλητής slurl η οποία ορίζεται σύμφωνα με την Εικόνα 4.5.

```
var url = location.href;
var slurl = (url.slice(url.indexOf('?') + 1, url.length));
```

Εικόνα 4.5. Η εξαγωγή της διεύθυνσης του URL του Second Life

Έτσι, η μεταβλητή `slurl` αποκτά τη διεύθυνση του URL του Second Life, αποκόπτοντας το κομμάτι της διεύθυνσης της κεντρικής σελίδας μετά τον χαρακτήρα «?». Υπενθυμίζεται ότι η κεντρική διεύθυνση που χρησιμοποιείται στον διαδικτυακό φυλλομετρητή (Web Browser) προκύπτει από τη σάρωση του QR κωδικού από το Second Life ή από τη χειροκίνητη τοποθέτηση του URL μετά τη διεύθυνση της HTML σελίδας στην οποία έχει τοποθετηθεί και ο χαρακτήρας «?». Το ίδιο URL θέλει να χρησιμοποιήσει και η μεταβλητή `slurl` και για αυτόν το λόγο αντιστρέφει αυτόματα την προηγούμενη διαδικασία.

Έτσι, επιστρέφοντας στα μέρη της νέας διεύθυνσης του νέου παραθύρου, στο δεύτερο μέρος της διεύθυνσης θα υπάρξει ο χαρακτήρας «?» ο οποίος θα χρησιμοποιείται από το Second Life για τον διαχωρισμό των συγκεκριμένων μερών της διεύθυνσης και την εξαγωγή πληροφορίας από τα δεδομένα.

Τέλος, στο τρίτο μέρος της διεύθυνσης τοποθετείται το κείμενο το οποίο περιλαμβάνει την επιλογή του χρήστη. Αυτό το κείμενο ουσιαστικά αποτελεί και τη μόνη πληροφορία που δέχεται εν τέλει το παιχνίδι Simon Says στο Second Life. Το συγκεκριμένο πεδίο μπορεί να λαμβάνει τις τιμές «RED», «GREEN», «BLUE», «YELLOW» και «START» τις οποίες δέχεται και φιλτράρει το αντικείμενο CONNECTOR του Simon Says στο Second Life.

4.3.1.3 ΤΟ ΠΕΔΙΟ IFRAME

Κανονικά, με το άνοιγμα νέου παραθύρου στον διαδικτυακό φυλλομετρητή κάθε φορά που ο χρήστης πατάει ένα κουμπί, το παιχνίδι Simon Says στον φυλλομετρητή θα ήταν αδύνατο. Διότι τα πολλαπλά παράθυρα που θα εμφανιζόντουσαν αλλά και η εναλλαγή στη νέα καρτέλα θα δημιουργούσε ασύμφορο περιβάλλον χρήστη και θα εξαφάνιζε τη λειτουργικότητα του εγχειρήματος.

Σε αυτό το πρόβλημα δίνει λύση το πεδίο `iframe` το οποίο αποτελεί εμφωλευμένη σελίδα HTML εντός άλλης σελίδας HTML .

```
<iframe name="win1" src="" width=0 height=0 frameborder=0></iframe>
```

Εικόνα 4.6. Το πεδίο `iframe`.

Το πεδίο `iframe` χρησιμοποιεί το νέο παράθυρο «win1» που δημιουργείται από την εντολή `window.open()` και του προσάγει μηδενικό πλάτος και μηδενικό ύψος, κάνοντας το ουσιαστικά αόρατο. Με αυτόν τον τρόπο αποφεύγεται η δημιουργία υπαρκτού νέου παραθύρου στον φυλλομετρητή και οι αρνητικές επιπτώσεις που προκαλεί αυτό. Παρόλα αυτά, νέο παράθυρο δημιουργείται, και μπορεί να διαπεραστεί από το Second Life, όμως απλά δε μπορεί οπτικά να ανιχνευθεί.

4.3.1.4 ΑΠΟΛΑΒΗ ΔΕΔΟΜΕΝΩΝ ΣΤΟ SECOND LIFE

Από τη στιγμή που το παραχθέν URL συνοδευτεί με τον χαρακτήρα «?» και στη συνέχεια τα δεδομένα του χρήστη από τον φυλλομετρητή, προκύπτει μια ζήτηση http request η οποία σκανδαλίζει (trigger) το συμβάν http_request (event) στο αντικείμενο CONNECTOR στο Second Life. Αυτός ο σκανδαλισμός δίνει το έναυσμα στον CONNECTOR να επεξεργαστεί την επικεφαλίδα που έχει παραχθεί στον φυλλομετρητή και μέσω της εντολής `llGetHTTPHeader(id, "x-query-string")` και της `llSetContentType(id, CONTENT_TYPE_HTML)` να αποκομίσει τα δεδομένα του χρήστη που υπάρχουν στην επικεφαλίδα του νέου παραθύρου στο πεδίο `iframe`. Η εντολή `llGetHTTPHeader(id, "x-query-string")` αποκομίζει τα δεδομένα που βρίσκονται στην επικεφαλίδα ενώ το πεδίο «x-query-string» ορίζει πως τα δεδομένα θα αποκτηθούν μετά τον χαρακτήρα «?» στην επικεφαλίδα. Ακόμα, η εντολή `llSetContentType(id, CONTENT_TYPE_HTML)` είναι απαραίτητη καθώς ορίζει πως τα δεδομένα που αποκτήθηκαν μέσω της επικεφαλίδας HTTP, προέρχονται από σελίδα HTML. Το πεδίο «id» και στις δύο εντολές ορίζει την ταυτότητα του κλειδιού της ζήτησης HTTP (HTTP request key)[5].

Να σημειωθεί πως η μέθοδος της HTTP request είναι η «GET» καθώς το Second Life σκοπεύει να αποκομίσει πληροφορία από τον Server[6]. Ο έλεγχος για τη μέθοδο είναι απαραίτητος για τον κώδικα του CONNECTOR στο Second Life και τον πετυχαίνει μέσω μιας δομής ελέγχου ροής `if`, όπως φαίνεται στην Εικόνα 4.7.

```
// If a homepage server request or a URL request comes in
http_request(key id, string method, string body)
{
    // If the request is an address
    if (method == URL_REQUEST_GRANTED)
    {
        // Show your new homepage address in chat window.
        url=body;
        //llOwnerSay("URL: " + url);
    }
    // When a GET request arrives
    else if (method == "GET")
    {
        //Data from our answer is contained in the HTTP header of our page
        //HTML page generates a new window(invisible window) in which's header, our answer it's written
        string result = llGetHTTPHeader(id,"x-query-string");//Data from our answer is being kept in the variable "result"
        llSetContentType(id, CONTENT_TYPE_HTML);
    }
}
```

Εικόνα 4.7. Το συμβάν `http_request` και ο έλεγχος μεθόδου

Επίσης, όπως φαίνεται, γίνεται ένας τυπικός έλεγχος επιβεβαίωσης της αποδοχής ζήτησης URL η οποία προσφέρει την επιλογή εμφάνισης του URL στον χρήστη. Ωστόσο, η διαχείριση των δεδομένων του χρήστη από τον φυλλομετρητή χρησιμοποιώντας HTTP επικοινωνία, συμβαίνει εντός της δομής ελέγχου `if` με τη μέθοδο να είναι «GET» (`method == "GET"`). Εκεί, χρησιμοποιώντας τις συναρτήσεις που προαναφέρθηκαν, τα δεδομένα του χρήστη αποθηκεύονται στη μεταβλητή `result`. Σε εκείνη τη στιγμή, τα δεδομένα του χρήστη είναι τύπου `string` και είναι εύκολα επεξεργάσιμα.

Στη συνέχεια, γίνεται έλεγχος της μεταβλητής `result` η οποία φέρει και τα δεδομένα του χρήστη. Αν η μεταβλητή είναι κενή, τότε θεωρείται πως το URL δεν έχει ακόμα χρησιμοποιηθεί, και για αυτό

ανοίγει ένα παράθυρο του εσωτερικού φυλλομετρητή του Second Life, εντός του οποίου υπάρχουν οι οδηγίες προς τον χρήστη να σαρώσει τον QR κωδικό ενώ επίσης η διεύθυνση σε αυτό το παράθυρο είναι το παραχθέν URL το οποίο ο χρήστης μπορεί να αντιγράψει και να το χρησιμοποιήσει ποικιλοτρόπως.

4.3.1.5 QR CODE

Ο QR κωδικός δημιουργείται δυναμικά, με τη μορφή της Εικόνας 3.32, χρησιμοποιώντας τη διεύθυνση της υπηρεσίας δημιουργίας QR κωδικών «<https://api.qrserver.com/v1/create-qr-code/?data=>». Αυτή η διεύθυνση, χρησιμοποιούμενη μαζί με την διεύθυνση που είναι επιθυμητό να κωδικοποιηθεί, δημιουργεί ένα μοναδικό QR κωδικό. Εντός του Second Life προβάλλεται ως εικόνα κάτω από το γραμμοσκιασμένο κείμενο που υποδεικνύει τον χρήστη να σαρώσει τον κωδικό. Σημαντικό εδώ είναι να παρατηρηθεί πως επειδή το συγκεκριμένο κείμενο αλλά και ο QR κωδικός εμφανίζονται στον εσωτερικό φυλλομετρητή του Second Life, οι παραπάνω εντολές είναι γραμμένες σε γλώσσα HTML, εντός της εντολής `llHttpResponse()` όπως φαίνεται στην Εικόνα 4.8, όπου το νούμερο 200 αντιστοιχεί στον κωδικό κατάστασης της HTTP επικοινωνίας «OK» που ορίζει ότι η επικοινωνία ήταν επιτυχής[7]. Η μεταβλητή `baseurl` έχει τη διεύθυνση της HTML σελίδας και το «?»

```
if(result=="")
//Proper URL is our baseurl plus the url which is generated from the "llRequestURL()"
//baseurl+url navigates to the window of our custom HTML page which is built for this specific HTTP communication ONLY
//If the URL is released then the wanted HTML page will be changed but it will be generated automatically again
llHttpResponse(id,200,header +
"<br/>Take a QR code with your smartphone.<br/>"
"<img src='https://api.qrserver.com/v1/create-qr-code/?data=" + baseurl + url + "'/>");
```

Εικόνα 4.8. Browser του Second Life με τον παραχθέν QR κωδικό

4.3.1.6 ΔΙΑΧΕΙΡΙΣΗ ΤΩΝ ΔΕΔΟΜΕΝΩΝ ΑΠΟ ΤΟ SECOND LIFE

Εφόσον τα δεδομένα του χρήστη είναι αποθηκευμένα στη μεταβλητή `result` και εφόσον είναι γνωστές οι πιθανές τιμές που μπορεί να λάβει αυτή η μεταβλητή, δεν αποτελεί πρόβλημα η επεξεργασία των τιμών της. Έτσι, με πέντε διαφορετικές δομές ελέγχου `if`, πάντα εμφωλευμένες εντός της δομής ελέγχου για τη μέθοδο HTTP GET, γίνεται κατηγοριοποίηση των περιπτώσεων και αναλόγως της τιμής, αποστέλλεται στο κεντρικό αντικείμενο PARENT το αντίστοιχο μήνυμα μέσω του καναλιού C στο Second Life. Οι πιθανές περιπτώσεις είναι οι «RED», «GREEN», «BLUE», «YELLOW» και «START» όπως φαίνεται στην Εικόνα 4.9.

```
if(result=="RED")//If our choice is "RED", the word "RED" is sent to "parent_v8"
llSay(-5154789,"RED");
if(result=="BLUE")//If our choice is "BLUE", the word "BLUE" is sent to "parent_v8"
llSay(-5154789,"BLUE");
if(result=="GREEN")//If our choice is "GREEN", the word "GREEN" is sent to "parent_v8"
llSay(-5154789,"GREEN");
if(result=="YELLOW")//If our choice is "YELLOW", the word "YELLOW" is sent to "parent_v8"
llSay(-5154789,"YELLOW");
if(result=="START")//If we want to start the game and pressed "START", the word "START" is sent to "parent_v8"
llSay(-5154789,"START");
```

Εικόνα 4.9. Η κατηγοριοποίηση των δεδομένων

4.3.2 ΔΕΔΟΜΕΝΑ ΑΠΟ SECOND LIFE

Εάν ο Web Server μπορούσε μόνο να στείλει δεδομένα στο Second Life και δεν υπήρχε επικοινωνία διπλής κατεύθυνσης, τότε ο Web Browser θα αποτελούσε ένα απλό χειριστήριο το οποίο θα μπορούσε να χρησιμοποιηθεί μόνο μπροστά από οθόνη συσκευής που εκτελούσε το Simon Says στο Second Life. Όμως, ο στόχος αυτού του εγχειρήματος αποτελούσε η διπλής κατεύθυνσης επικοινωνία μεταξύ του Second Life και μιας εξωτερικής συσκευής. Αυτό επιτεύχθηκε με τη χρησιμοποίηση PHP και AJAX εργαλείων.

Τα δεδομένα που πρέπει να διαβιβαστούν μέσω του Web Server στον χρήστη, έχουν τη μορφή κειμένου «RGRB». Αυτά τα δεδομένα αποτελούν τη ζητούμενη χρωματική ακολουθία που πρέπει να εισάγει ο χρήστης για να προχωρήσει στο παιχνίδι. Οπότε αποτελεί ζητούμενο, η παραλαβή των δεδομένων από την HTML σελίδα και η εξαγωγή του καθενός χρώματος ώστε να είναι δυνατή η αναπαραγωγή της χρωματικής ακολουθίας στον χρήστη.

4.3.2.1 ΑΠΟΣΤΟΛΗ ΑΠΟ SECOND LIFE ΣΕ PHP ΑΡΧΕΙΟ

Αρχικά, το αντικείμενο CONNECTOR παραλαμβάνει τα δεδομένα από το κεντρικό αντικείμενο PARENT και τα αποθηκεύει στη μεταβλητή PHP_DATA. Ωστόσο, η συγκεκριμένη μεταβλητή περιέχει τη διεύθυνση του PHP αρχείου που βρίσκεται στον Host Server, ενώ επιπλέον προστίθεται η κατάληξη «?test=». Με αυτόν τον τρόπο η μεταβλητή PHP_DATA περιέχει τη διεύθυνση «http://siwonsimon.000webhostapp.com/sl.php?test=» η οποία αποτελεί το περιεχόμενο της μεταβλητής PHP_URL και ακόμη, μετά τον χαρακτήρα «=», τοποθετείται το κείμενο με τη χρωματική ακολουθία που έχει παραχθεί. Ένα τέτοιο παράδειγμα θα μπορούσε να είναι το «http://siwonsimon.000webhostapp.com/sl.php?test=RGRB».

Επιπλέον, πέρα από τη χρωματική ακολουθία, μπορεί να αποσταλεί ο χαρακτήρας «Ο», εκ του GAME OVER, ο οποίος αποτελεί το μήνυμα λάθους προς τον χρήστη και την ένδειξη ότι ο τωρινός γύρος του παιχνιδιού τερματίζεται.

```
//Communication with "parent_v8" for gaining the wanted sequence of colors
if (channel == -5154789){
    PHP_URL= "http://siwonsimon.000webhostapp.com/sl.php?test=";
    PHP_DATA= PHP_URL + message;//Color sequence is a string in the variable "message"(e.g. "RRYBG")
    //R stands for RED, G stands for GREEN, B stands for BLUE, Y stands for YELLOW, O stands for GAME OVER
    $HTTPREQUEST(PHP_DATA,[""]);//Sending the wanted sequence into a php file
    //in which is been processed and saved to a TXT file
}
```

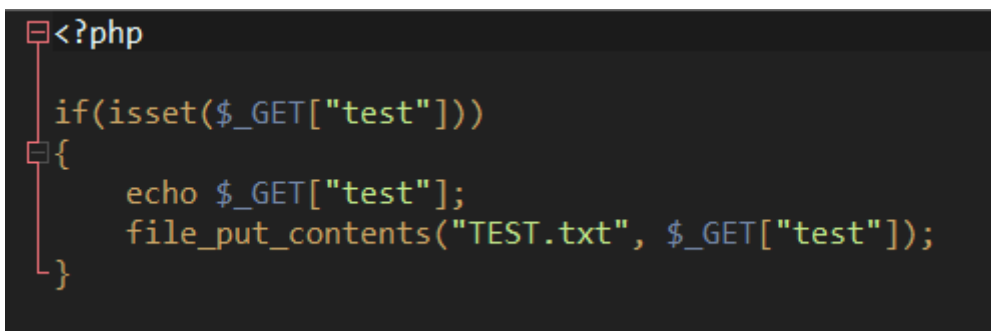
Εικόνα 4.10. Η απόκτηση των δεδομένων από το αντικείμενο PARENT.

Τα δεδομένα από το αντικείμενο PARENT μεταβιβάζονται στη μεταβλητή PHP_DATA μέσω της μεταβλητής message χρησιμοποιώντας το συμβάν listen.

Στη συνέχεια, χρησιμοποιώντας HTTP επικοινωνία, διαπερνιέται το PHP αρχείο το οποίο χρησιμοποιεί τα δεδομένα της διεύθυνσης που είναι αποθηκευμένη στη μεταβλητή PHP_DATA, μετά τον χαρακτήρα «?».

4.3.2.2 ΕΠΕΞΕΡΓΑΣΙΑ ΣΕ PHP

Το PHP αρχείο που βρίσκεται αποθηκευμένο στον Host Server, το οποίο θα πρέπει υποχρεωτικά να βρίσκεται στον ίδιο Host με το αρχείο HTML, παραλαμβάνει τα δεδομένα τα οποία στάλθηκαν από το Second Life, τα επεξεργάζεται με μια απλή συνάρτηση η οποία χρησιμοποιεί τη μεταβλητή test και εξάγει τα δεδομένα σε ένα αρχείο κειμένου (txt).



```
<?php
if(isset($_GET["test"]))
{
    echo $_GET["test"];
    file_put_contents("TEST.txt", $_GET["test"]);
}
```

Εικόνα 4.11. Το αρχείο PHP.

Εντός του αρχείου PHP, ελέγχεται αν η μεταβλητή test έχει τιμή, ενώ αναπαράγει την τιμή της μεταβλητής και ταυτόχρονα την εγγράφει στο αρχείο TEST.txt. Κάθε φορά που εγγράφονται δεδομένα στο συγκεκριμένο αρχείο, τα παλιότερα δεδομένα διαγράφονται. Το συγκεκριμένο αρχείο κειμένου δημιουργείται αυτόματα στον Host που βρίσκεται το php αρχείο ενώ αν υπάρχει αρχείο με το ίδιο όνομα, το διαγράφει και στη θέση του δημιουργεί το προαναφερθέν.

Το αρχείο PHP επιλέχθηκε να μην τοποθετηθεί εμφωλευμένα εντός της σελίδας HTML, εν αντιθέσει με όλα τα υπόλοιπα εργαλεία που χρησιμοποιήθηκαν, καθώς η αποτελεσματικότητα και η καθυστέρηση που δημιουργούνταν, δεν ήταν οι επιθυμητές.

4.3.2.3 ΠΡΟΣΠΑΘΕΙΑ ΑΝΑΓΝΩΣΗΣ ΔΕΔΟΜΕΝΩΝ ΜΕ PHP

Κατά τον σχεδιασμό της σελίδας HTML ήταν επιθυμητό η σελίδα να μην ανανεώνεται κάθε φορά που στον χρήστη έπρεπε να παρουσιαστούν νέα δεδομένα αλλά ούτε και ο χρήστης να μπει σε διαδικασία να κάνει χειροκίνητες ενέργειες ώστε να φορτωθούν τα δεδομένα στη σελίδα HTML. Ο πρώτο τρόπος που υλοποιήθηκε όριζε έναν PHP κώδικα το οποίο αποτελούσε μέρος της σελίδας HTML, που όμως χρειαζόταν να διατηρεί τη σελίδα HTML σε μόνιμη ανανέωση[8]. Με αυτόν τον τρόπο η σελίδα είχε

πρόσβαση σε νέα δεδομένα από το Second Life αυτόματα. Παρόλα αυτά, μια μόνιμη ανανέωση της σελίδας για αυτόματη φόρτιση με νέα δεδομένα, απαιτεί σταθερή σύνδεση στο διαδίκτυο χωρίς την παραμικρή ανωμαλία ενώ και ο φυλλομετρητής που έχει φορτώσει τη σελίδα θα πρέπει να ανανεώνει τη σελίδα χωρίς την παραμικρή καθυστέρηση. Κάτι τέτοιο φαντάζει ουτοπικό γιατί σε όλες τις συνδέσεις ίντερνετ υπάρχουν διακυμάνσεις ενώ όλοι οι φυλλομετρητές, αργά ή γρήγορα εμφανίζουν μικρές ανωμαλίες στην απόδοση τους. Αυτές οι ανωμαλίες δεν είναι αρκετές για να ζημιώσουν την εμπειρία χρήσης όμως αποδεικνύονται υπεραρκετές για να σταματήσει η σελίδα PHP να φορτώνει δεδομένα. Για αυτόν τον λόγο εγκαταλείφθηκε αυτή η κατεύθυνση και προτιμήθηκε η υλοποίηση με AJAX (Asynchronous JavaScript And Xml).

4.3.2.4 ΧΡΗΣΗ AJAX

Η AJAX αποτελεί πακέτο τεχνικών η οποία επέτρεψε στη σελίδα HTML να φορτώνει στο παρασκήνιο, δεδομένα από το αρχείο κειμένου TEST.txt χωρίς να χρειάζεται ανανέωση της σελίδας. Αποτελεί το κυριότερο εργαλείο για την ανάγνωση των δεδομένων από το Second Life καθώς χωρίς αυτό, δε θα ήταν δυνατή η υλοποίηση του project. Για τη συγκεκριμένη εργασία επιλέχθηκε η υλοποίηση με AJAX που προσφέρει η w3schools[9] με τη χρήση του XMLHttpRequest αντικειμένου για την επικοινωνία. Σημειώνεται ότι για τη συγκεκριμένη υλοποίηση επιλέχθηκε η σύγχρονη επικοινωνία καθώς η φύση του παιχνιδιού θα προϋπέθετε πολλαπλούς επιπλέον χειριστές (handlers). Όμως, για να μπορούν τα γραφικά που δημιουργήθηκαν να λειτουργούν επαρκώς αλλά και οι κλήσεις συναρτήσεων από άλλες συναρτήσεις εντός της JavaScript να μπορούν να εκτελεστούν, επιλέχθηκε η σύγχρονη εκτέλεση. Ωστόσο, το μεγαλύτερο πρόβλημα αποτελούσαν οι κλήσεις των συναρτήσεων από άλλες συναρτήσεις. Εξαιτίας του τρόπου ανάγνωσης της χρωματικής ακολουθίας αλλά και της ανάγκης να υπάρχει διαρκής ανάγνωση για δεδομένα ώστε μήπως εμφανιστεί ο χαρακτήρας «Ο» ο οποίος θα υποδήλωνε ότι το γύρος τελείωσε, κλήσεις AJAX θα έπρεπε να συμβαίνουν διαρκώς, μετά το κάθε πάτημα κουμπιού από τον παίκτη. Αυτές οι κλήσεις, σε περίπτωση ασύγχρονης επικοινωνίας θα δυσχέραιναν τον σχεδιασμό και την εκτέλεση του κώδικα αφού θα έπρεπε να αλλάξει τελείως η φιλοσοφία πίσω από το παιχνίδι.

4.3.2.5 ΑΝΑΓΝΩΣΗ ΔΕΔΟΜΕΝΩΝ ΣΤΗΝ HTML ΣΕΛΙΔΑ ΜΕ AJAX

Για να μπορεί η HTML σελίδα να φορτώνει νέα δεδομένα από το αρχείο κειμένου TEST.txt, χρησιμοποιεί το αντικείμενο XMLHttpRequest() με το οποίο ορίζει(ζητάει, request) μια καινούργια HTTP επικοινωνία η οποία ανανεώνει μόνο μέρος της σελίδας και όχι ολόκληρη τη σελίδα. Η τιμή (κλειδί, key) της HTTP επικοινωνίας αποθηκεύεται σε μια μεταβλητή xmlhttp. Στη συνέχεια,

χρησιμοποιώντας το key της επικοινωνίας, μπορεί να διαπεραστεί το αρχείο κειμένου TEST.txt με την εντολή xmlhttp.open("GET", "TEST.txt", false) ενώ με την εντολή xmlhttp.send() τα δεδομένα καταφτάνουν στον κώδικα JavaScript εντός της HTML (ουσιαστικά στην AJAX). Κατά το άνοιγμα του αρχείου κειμένου, χρησιμοποιείται η «GET» μέθοδος, ορίζεται το όνομα του αρχείου ενώ επίσης διευκρινίζεται η σύγχρονη μορφή της επικοινωνίας (false για σύγχρονη, true για ασύγχρονη).

Εδώ είναι σημαντικό να σημειωθεί πως η τεχνική AJAX, τουλάχιστον μέχρι τώρα, χρησιμοποιείται αποκλειστικά σε περιβάλλον κοινού Host (same-origin policies)[10] αν και υπάρχουν ορισμένα προγραμματιστικά εργαλεία που επιτρέπουν μερική επίλυση του συγκεκριμένου προβλήματος (πχ CORS)[11]. Αυτό σημαίνει πως δε μπορεί να επιτευχθεί επικοινωνία με AJAX εκτός του domain που χρησιμοποιεί η κεντρική σελίδα HTML.

```
function loadDoc() {  
    var xmlhttp = new XMLHttpRequest();  
    xmlhttp.open("GET", "TEST.txt", false); //AJAX call for synchronous data transfer  
    xmlhttp.send(); //Asynchronous wouldn't aide us cause we would have to add multiple handlers (time or not) for smooth running  
    DATA = xmlhttp.responseText;  
    DataLength = DATA.length;  
}
```

Εικόνα 4.12. Απόκτηση δεδομένων μέσω AJAX.

Τέλος, τα δεδομένα αποθηκεύονται στη μεταβλητή DATA μέσω της εντολής xmlhttp.responseText η οποία και θα χρησιμοποιείται από τον κώδικα JavaScript στη συνέχεια για την προβολή της ακολουθίας ή τον έλεγχο τερματισμού.

Οι βασικοί αυτοί υπολογισμοί ανήκουν στη συνάρτηση JavaScript με όνομα loadDoc() η οποία δέχεται αλλά και επεξεργάζεται τα δεδομένα από το Second Life.

4.3.2.6 ΕΠΕΞΕΡΓΑΣΙΑ ΔΕΔΟΜΕΝΩΝ ΣΤΗΝ HTML ΣΕΛΙΔΑ

Τα δεδομένα χρησιμοποιούνται από τη σελίδα HTML μέσω της μεταβλητής DATA. Ο κώδικας JavaScript στη συνέχεια ελέγχει το κάθε χαρακτήρα της ακολουθίας που έχει αποσταλεί μέσω της εντολής DATA.charAt(i), όπου i θετικός αριθμός μικρότερος ή ίσος του μήκους του DATA, και ανάλογα τη τιμή του, αναβοσβήνει το αντίστοιχο χρωματιστό κουμπί για 500ms (0.5 δευτερόλεπτα). Όπως έχει προαναφερθεί, οι τιμές που μπορεί να λάβει θα είναι μια από τις 5 περιπτώσεις, «R», «G», «B», «Y» και «O».

Επίσης, πριν εκτελεστεί το flash (αναβόσβημα), το πρόγραμμα ελέγχει αν τα δεδομένα που έχει αποκομίσει, είναι τα ίδια με αυτά που αποκόμισε πριν, μέσω της μεταβλητής previousDATA. Αν συμβαίνει κάτι τέτοιο, τότε δεν εκτελείται η προβολή της ακολουθίας καθώς αυτό σημαίνει πως δεν υπήρξε αλλαγή της ακολουθίας στο μεσοδιάστημα μεταξύ συνεχόμενων ελέγχων στο αρχείο κειμένου.

Δυστυχώς, αχρείαστοι έλεγχοι στο αρχείο κειμένου TEST.txt γίνονται κατ' εξακολούθηση οπότε θα πρέπει να υπάρχει και έλεγχος στη σελίδα HTML για το αν τα δεδομένα είναι καινούργια ή απλά είναι τα παλιά. Το συγκεκριμένο πρόβλημα θα το έλυne η ασύγχρονη λειτουργία του AJAX η οποία θα ενημέρωνε τη σελίδα HTML μόνο όταν υπήρχε αλλαγή των δεδομένων στο αρχείο κειμένου TEST.txt. Όμως, όπως προαναφέρθηκε, η ασύγχρονη λειτουργία είτε θα «πάγωνε» τη σελίδα HTML όσο θα περίμενε για δεδομένα, κάνοντας το περιβάλλον και την εμπειρία χρήστη, μη βέλτιστη, είτε θα επέτρεπε στον χρήστη να προχωράει στο παιχνίδι χωρίς αυτός όμως να μη γνωρίζει πως οι κινήσεις του είναι λανθασμένες ή ότι η χρωματική ακολουθία άλλαξε. Για να δουλέψει ένα τέτοιο εγχείρημα θα έπρεπε να υπάρχει πλήθος χειριστών και περιοριστών (μεταβλητών) που θα επέτρεπαν την ομαλή λειτουργία. Παρόλα αυτά, επιλέχθηκε η σύγχρονη μέθοδος με την προσθήκη χρονικών καθυστερήσεων (ελάχιστης διάρκειας) που επιτρέπουν στην ομαλότερη λειτουργία που επιτεύχθηκε.

```
var start,end;
var i;
if (DATA.charAt(0) == ""){
    setTimeout(function() {loadDoc();},500);
}
if (previousDATA == DATA){
    setTimeout(function() {loadDoc();},500);
}
else{
    previousDATA = DATA;
    for (i = 0; i < DataLength; i++) {
        start = 1200 * i; //Start and End Time Calculations
        end = start + 500;
        if (DATA.charAt(i) == "R"){
            R_MOVE(start,end);
        }
        else if (DATA.charAt(i) == "G"){
            G_MOVE(start,end);
        }
        else if (DATA.charAt(i) == "B"){
            B_MOVE(start,end);
        }
        else if (DATA.charAt(i) == "Y"){
            Y_MOVE(start,end);
        }
    }
}
```

Εικόνα 4.13. Έλεγχος δεδομένων και εκτέλεση flashing κουμπιών.

Τέλος, η προβολή των ζητούμενων κουμπιών γίνεται με διαφορά 700ms μέσω των προσαρμοσμένων συναρτήσεων R_MOVE, G_MOVE, B_MOVE και Y_MOVE οι οποίες αλλάζουν το χρώμα του κάθε κουμπιού, στη φωτεινή τους μορφή.

```
//=====Function who makes RED Button blink when the PHP sent data is "R"=====//
function R_MOVE(start_time,end_time) {
    if (document.getElementById("PRESSR") != null) {
        setTimeout(function() {
            document.getElementById('PRESSR').style.backgroundColor = '#FF4433';
        }, start_time);
        setTimeout(function() {
            document.getElementById('PRESSR').style.backgroundColor = '#BF3326';
        }, end_time);
    }
}
```

Εικόνα 4.14. Η συνάρτηση που είναι υπεύθυνη για το blinking (flashing) του κόκκινου κουμπιού.

Αυτή η εναλλαγή, διαρκεί για 500ms και ορίζεται από τις μεταβλητές start και end που καθορίζουν πότε το κάθε κουμπί θα ξεκινήσει την εναλλαγή χρώματος και πότε θα τελειώσει, παρομοιάζοντας flash. Επειδή, η δομή επανάληψης, αν και εκτελεί μια συνάρτηση σε κάθε επανάληψη, εκτελείται ταχύτατα, χρειάστηκε να εισαχθούν οι χρονικές μεταβλητές start και end. Σε αυτήν τη χρονική επεξεργασία, η αρχή του χρόνου βρίσκεται στην προβολή του πρώτου χρώματος. Από εκεί και πέρα το κάθε επόμενο κουμπί ανάβει μετά από 700ms από όταν κλείσει το προηγούμενο.

```
else if (DATA.charAt(i) == "O"){
    setTimeout(function() {document.getElementById("button2").innerHTML = "GAME OVER";}, 500);
    if(side_step < score){
        if ((side_score == 1)&&(side_step<2)){
            setTimeout(function() {document.getElementById("button2").innerHTML = "YOUR SCORE: 0";}, 2000);
        }
        else{
            setTimeout(function() {document.getElementById("button2").innerHTML = "YOUR SCORE:"+ score;}, 2000);
        }
    }
    else{
        setTimeout(function() {document.getElementById("button2").innerHTML = "YOUR SCORE:"+ (score-1);}, 2000);
    }
    setTimeout(function() {document.getElementById('button2').style.transform = 'translateY(-50%)';}, 3500);
    setTimeout(function() {document.getElementById('button2').style.visibility = 'hidden';}, 3500);
    setTimeout(function() {document.getElementById('button1').style.transform = 'translateY(50%)';}, 3800);
    setTimeout(function() {document.getElementById('button1').style.visibility = 'visible';}, 3800);
    previousDATA=0;
    step=0;
    setTimeout(function() {zero();},3800);
}
```

Εικόνα 4.15. Περίπτωση εμφάνισης χαρακτήρα «O» στα δεδομένα υποδεικνύοντας «GAME OVER».

Τέλος, στην περίπτωση που καταφθάσει ως δεδομένο, ο χαρακτήρας «O», γίνεται υπολογισμός του σκορ ενώ μπαίνει σε λειτουργία και το animation του κουμπιού «START GAME». Ουσιαστικά, δεν προκύπτει κάποιου είδους επαναφορά στη σελίδα εκτός από τον μηδενισμό ορισμένων μεταβλητών με τη χρήση της συνάρτησης zero().

```
function zero(){
    score=0;
    side_step=0;
    side_score=0;
}
```

Εικόνα 4.16. Η συνάρτηση zero.

4.4 ΠΡΩΤΟΚΟΛΛΑ ΕΠΙΚΟΙΝΩΝΙΑΣ ΚΑΙ ΠΕΡΙΟΡΙΣΜΟΙ

Καθ' όλη τη διαδρομή των δεδομένων από το Second Life προς τον Web Server και από τον Web Server στο Second Life, τα δεδομένα μεταφέρονται με ποικίλους τρόπους, ενώ αν και διατηρούν την αρχική μορφή κειμένου τους, η εναλλαγή πολλαπλών πλατφόρμων και ο εμπλουτισμός τους από νέα δεδομένα χρήστη αγγίζουν πολλαπλές ιδιότητες και πρωτόκολλα που έπρεπε να ληφθούν υπόψιν κατά τη διάρκεια της εργασίας. Παρόλα αυτά, το μέγεθος των δεδομένων, αν και σε πολλές άλλες εφαρμογές θα ήταν το κυριότερο πρόβλημα και η μεγαλύτερη εστία προσοχής κατά την επεξεργασία των δεδομένων, στο συγκεκριμένο project απασχόλησε ελάχιστα την παραγωγή κώδικα αφού εκ των προτέρων υπήρχε η μέριμνα για ταχεία και αποτελεσματική μεταφορά δεδομένων, που απαιτούσε μικρά κομμάτια σε μέγεθος, δεδομένων. Ωστόσο, ακόμα κι εδώ, μπορούν να υπάρξουν ακραίες περιπτώσεις που αγγίζουν τα όρια της λειτουργίας του project και μπορούν δυνητικά να δημιουργήσουν ίσως σε σπάνιες περιπτώσεις, πρόβλημα.

4.4.1 ΠΡΩΤΟΚΟΛΛΑ ΚΑΙ ΠΕΡΙΟΡΙΣΜΟΙ ΣΤΟ SECOND LIFE

Στο Second Life τα δεδομένα δέχονται μεταφορά μέσω καναλιών επικοινωνίας εντός του Second Life, παραλαμβάνονται με HTTP σύνδεση από Web Server ενώ τέλος, αποστέλλονται στον Web Browser. Η πύλη (port) που χρησιμοποιείται από τον Server του Second Life για όλες τις επικοινωνίες μέσω HTTP στις οποίες λαμβάνει δεδομένα το Second Life, είναι η «12046» [14].

4.4.1.1 ΚΑΝΑΛΙΑ ΕΠΙΚΟΙΝΩΝΙΑΣ

Στο Second Life τα δεδομένα διαβιβάζονται μέσω custom-made καναλιών (channel) τα οποία τα αποστέλλουν ως κείμενο στα αντικείμενα ή στους χρήστες που βρίσκονται εντός εμβέλειας

Function	Distance
llWhisper	10 metres
llSay	20 metres
llShout	100 metres
llRegionSay	region-wide
llRegionSayTo	region-wide

Πίνακας 4.1. Εύρος ακρόασης διάφορων συναρτήσεων εντός του Second Life

Τα συγκεκριμένα κανάλια μπορούν να είναι είτε θετικά είτε αρνητικά ενώ το εύρος των καναλιών είναι από -2147483648 έως 2147483647.

Το μέγεθος των μηνυμάτων μπορεί να είναι μέχρι 1023 byte στα θετικά και 254 byte στα αρνητικά. Ο συγκεκριμένος περιορισμός αποτελεί αναγνωρισμένο bug (προγραμματιστικό λάθος)[12]. Κάθε script μπορεί να «ακούει» έως και 65 κανάλια ταυτοχρόνως[13].

4.4.1.2 ΠΑΡΑΛΑΒΗ ΔΕΔΟΜΕΝΩΝ

Για την παραλαβή των δεδομένων από τον Web Server, ο κώδικας στο Second Life χρησιμοποιεί το συμβάν (event) `http_request`. Όμως, τα δεδομένα προκύπτουν από την ανάγνωση της επικεφαλίδας που προκύπτει στην HTTP σύνδεση. Αυτός διέπεται μέσω της εντολής `llGetHTTPHeader(id, "x-query-string")`. Ο Header (επικεφαλίδα) περιορίζεται στα 255 byte ενώ είναι διαθέσιμος μόνο για 30 δευτερόλεπτα από τη κλήση του event `http_request[5]` το οποίο σκανδαλίζεται από την αίτηση που δημιουργεί η HTML σελίδα με τη δημιουργία νέου παραθύρου. Σημειώνεται πως τα δεδομένα που προκύπτουν από την επικεφαλίδα αποθηκεύονται σε μεταβλητή string ενώ για να είναι προσπελάσιμα θα πρέπει να οριστούν ως δεδομένα HTML με την εντολή `llSetContentType(id, CONTENT_TYPE_HTML)`. Για να χρησιμοποιηθεί το `CONTENT_TYPE_HTML` θα πρέπει η `http_request` να εκτελείται εντός Second Life, ο χρήστης (συνδεδεμένος στον Second Life Client που βλέπει τη σελίδα, δηλαδή το παραγόμενο url) να είναι ο ιδιοκτήτης του αντικειμένου και τέλος, ο χρήστης να είναι συνδεδεμένος στην ίδια περιοχή του Second Life (region) που βρίσκεται και το αντικείμενο. Τέλος, σημαντικό είναι στο παραγόμενο URL του Second Life να τοποθετείται ο χαρακτήρας «?» πριν από κάποια απάντηση ενώ θα πρέπει πριν τον χαρακτήρα «?» να υπάρχει κάθετος χαρακτήρας «/». Σε περίπτωση απώλειας του το status της HTTP σύνδεσης ορίζεται ως 500 «Server Error code»[14].

4.4.1.3 ΑΠΟΣΤΟΛΗ ΔΕΔΟΜΕΝΩΝ

Το Simon Says στο Second Life στέλνει δεδομένα προς τον Web Server μέσω HTTP request, όχι στο παραγόμενο URL που χρησιμοποιήθηκε προηγουμένως, αλλά κατευθείαν στην PHP σελίδα που βρίσκεται στον Web Server. Έτσι, αυτή η HTTP επικοινωνία είναι διαφορετική από την προηγούμενη καθώς ο Server σε αυτήν την περίπτωση είναι ο Web Server και όχι ο Server του Second Life. Η εντολή που χρησιμοποιείται είναι η `llHTTPRequest(string url, list parameters, string body)`, όπου string url είναι ένα έγκυρο HTTP/HTTPS URL. Στη συγκεκριμένη περίπτωση, το sting url είναι η μεταβλητή `PHP_DATA` που περιέχει τη διεύθυνση της σελίδας PHP μαζί με τη μεταβλητή και την τιμή της που θα αποσταλούν στη σελίδα. Η λίστα παραμέτρων (list parameters) είναι κενή οπότε χρησιμοποιείται η

προκαθορισμένη (default) μέθοδος HTTP, «GET». Προσοχή απαιτείται στην τιμή της μεταβλητής της PHP σελίδας που αποστέλλεται με την διεύθυνση στην HTTP επικοινωνία καθώς κενοί χαρακτήρες, χαρακτήρες ελέγχου και άλλοι που δεν επιτρέπονται στις διευθύνσεις URL, θα προκαλούν σφάλμα εκτέλεσης (run time error). Οι χαρακτήρες που επιτρέπονται είναι σετ utf-8[15]. Επίσης, οι κλήσεις HTTP θα πρέπει να πραγματοποιούνται εντός 60 δευτερολέπτων, αλλιώς η κλήση θα χαθεί και το status code (κωδικός κατάστασης) της HTTP απάντησης θα είναι «499». Οι συνολικοί χαρακτήρες στη διεύθυνση URL που διέπεται μπορεί να έχει μέγιστο μήκος 253 χαρακτήρων [15].

Σημαντικό είναι να ειπωθεί εδώ πως επιτρέπονται 25 κλήσεις `llHttpRequest()` εντός 20 δευτερολέπτων για κάθε αντικείμενο ενώ επιτρέπονται 1000 κλήσεις `llHttpRequest()` εντός 20 δευτερολέπτων για κάθε ιδιοκτήτη. Οι συγκεκριμένες τιμές είναι πιθανό να αλλάξουν μελλοντικά καθώς αποτελούν αντικείμενο ανάπτυξης για το Second Life. Αν επιτευχθεί το ανώτατο όριο κλήσεων(throttle blocking), τότε η επιστροφή της κλήσης `llHttpRequest()` θα είναι η «NULL_KEY». Σε τέτοια περίπτωση ο χρόνος αναμονής για επαναφορά σε κανονικές τιμές είναι τα 40 δευτερόλεπτα[15].

4.4.1.4 ΔΕΣΜΕΥΣΗ URL

Συνιστάται έντονα η απελευθέρωση των διευθύνσεων URL πριν από τη δημιουργία ενός νέου ή μετά τον τερματισμό του χρόνου παιχνιδιού. Αυτό συμβαίνει επειδή οι διευθύνσεις URL τείνουν να υπάρχουν ακόμα και μετά το κλείσιμο του παιχνιδιού. Καταλαμβάνουν επίσης ένα φυσικό χώρο στη μνήμη μιας περιοχής (region) στο Second Life. Αυτό σημαίνει ότι εάν ένα in-world region έχει χωρητικότητα 1000 prim(τύπος αντικειμένου στη Second Life) τότε θα μπορούσε να γεμίσει με 1000 διευθύνσεις URL και ο ιδιοκτήτης του δεν θα ήταν σε θέση να χτίσει ή να γράψει τίποτα. Αυτό μπορεί να οδηγήσει στην απαγόρευση των χρηστών από ορισμένες περιοχές, αν δεν έχουν την ιδιοκτησία της γης. Η Second Life έχει προβλέψει αυτό το πρόβλημα και έχει υλοποιήσει μια αυτόματη απελευθέρωση URL. Ωστόσο, αυτή η έκδοση δεν λειτουργεί άμεσα και δεν έχει τέλεια απόδοση.



Important: Never ever forget to release a URL again which you have requested! URLs are region resources just like prims. If you take them all you can get into big trouble with the sim owner and/or estate managers.

Εικόνα 4.17. Μήνυμα που υπάρχει στην εγκυκλοπαίδεια του Second Life

Η εντολή `llGetFreeURLs()` μπορεί να χρησιμοποιηθεί ώστε να εμφανιστεί ο αριθμός των διαθέσιμων URL της περιοχής που προορίζονται για δωρεάν δέσμευση. Η χρησιμοποίηση της εντολής `llRequestURL()` είναι περιορισμένη, αν και δεν υπάρχει ακριβή μέτρηση για το πόσες χρήσεις

επιτρέπονται, συνίσταται αυτές να είναι λιγότερες από 5 σε σύντομο χρονικό διάστημα. Η χρήση της εντολής μετά από απόρριψη εξυπηρέτησης θα είναι διαθέσιμη μετά από 1 δευτερόλεπτο. Προτείνεται να χρησιμοποιείται συνάρτηση παύσης (IISleep) για 0.6 δευτερόλεπτα ή και περισσότερο, ανάμεσα σε συνεχόμενες κλήσεις. Επιπλέον, κάθε φορά που μια περιοχή επανεκκινείται, όλα τα URL αποδεσμεύονται και καθίστανται άχρηστα[16].

Για αποδέσμευση των URL χρησιμοποιείται η εντολή IIReleaseURL() ενώ στις παρακάτω περιπτώσεις, τα URL αποδεσμεύονται αυτόματα.

- Όταν η περιοχή επανεκκινηθεί ή τεθεί εκτός σύνδεσης.
- Όταν το πρόγραμμα (script) που κατέχει τις διευθύνσεις URL επαναρυθμίζεται (reseted) ή μεταγλωττίζεται (compiled).
- Όταν το αντικείμενο που περιέχει το πρόγραμμα (script) διαγράφεται ή μεταφέρεται στο απόθεμα.

Παρόλα αυτά, όπως έχει ήδη προαναφερθεί, η αυτόματη αποδέσμευση των URL δεν είναι 100% αποτελεσματική, όπως έχει αναφερθεί από κάποιους χρήστες.

4.4.2 ΠΡΩΤΟΚΟΛΛΑ ΚΑΙ ΠΕΡΙΟΡΙΣΜΟΙ ΣΤΟΝ WEB SERVER

Στον Web Server αποστέλλονται και φθάνουν δεδομένα προς και από το Second Life μέσω μιας σελίδας HTML και μιας σελίδας PHP. Για αυτήν την επικοινωνία χρησιμοποιείται HTTP επικοινωνία. Αλλά, ταυτόχρονα επιτελείται ακόμη μια επικοινωνία, αυτή μεταξύ της σελίδας HTML και της σελίδας PHP. Εκεί, χρησιμοποιείται το σύνολο εργαλείων AJAX που βασίζεται στη φόρτωση δεδομένων στην HTML σελίδα, χωρίς την ανανέωση της.

4.4.2.1 ΕΠΙΚΟΙΝΩΝΙΑ HTML ΠΡΟΣ SECOND LIFE

Η σελίδα HTML επικοινωνεί με το Second Life μέσω HTTP επικοινωνίας. Η HTML σελίδα δημιουργεί ένα νέο παράθυρο περιήγησης και εισάγει στην επικεφαλίδα του, το παραχθέν URL του Second Life και επιπλέον την επιλογή του χρήστη. Αυτή η κλήση στον Server του Second Life σκανδαλίζει το πρόγραμμα στο Second Life, το οποίο αναγιγνώσκει την επικεφαλίδα. Οι περιορισμοί στην μεταφορά της επικεφαλίδας καλύφθηκαν από την παράγραφο 4.4.1.3 καθώς η HTTP επικοινωνία εκκινείται από το Second Life και τον Second Life Server.

Παρόλα αυτά, το νέο παράθυρο που δημιουργείται, τοποθετείται σε μια περιοχή iframe η οποία φέρει έναν κύριο περιορισμό όσον αφορά την ασφαλή περιήγηση.

4.4.2.2 IFRAME και SSL

Στο Second Life υπάρχει ξεχωριστή εντολή που παράγει URL με SSL πιστοποιητικό και αυτή είναι η `IIRequestSecureURL()`. Όμως, ενώ αυτή η εντολή δουλεύει και παράγεται URL με HTTPS της μορφής `"https://sim10186.agni.lindenlab.com:12043/cap/469d084a-e1ab-2e35-6e6d-95c3d6f464b0"` το URL αυτό δεν φέρει ενημερωμένο πιστοποιητικό ασφαλείας.

Αυτό έχει σαν αποτέλεσμα όλοι οι Web Browser να εντοπίζουν την κατάληξη `https` εντός του `iframe` στην HTML σελίδα αλλά μόλις η σελίδα φορτώνεται, να μην έχουν τη δυνατότητα να εντοπίσουν ισχύον πιστοποιητικό και άρα να το θεωρούν απειλή. Έτσι όλοι οι Web Browser που δοκιμάστηκαν (Mozilla, Chrome, Edge) δεν επιτρέπουν την φόρτιση της σελίδας αλλά και την επικοινωνία με το Second Life. Το πρόβλημα προκαλείται με την επίτευξη μονομερούς πιστοποίησης SSL για τη σελίδα HTML ενώ ταυτοχρόνως το URL του Second Life δε φέρει πιστοποιητικό.

Σε αυτό φέρει ευθύνη η αντιμετώπιση των πεδίων `iframe` από τους περιηγητές διαδικτύου. Η μη ασφαλής εκτέλεση `iframe` δημιουργεί προβλήματα στους Browsers και για αυτό τις απορρίπτουν απολύτως αν υπάρχει η παραμικρή αμφιβολία ασφάλειας ενώ ακόμη κι αν η σελίδα HTML είναι HTTPS, το `iframe` δε μπορεί να είναι HTTP[17]. Για αυτή την αδυναμία παροχής `https` σύνδεσης από το Second Life ευθύνεται η μητρική εταιρεία της πλατφόρμας, η Linden, η οποία ακόμα και στην εγκυκλοπαίδεια του Second Life, δεν έχει ενημερώσει τα πιστοποιητικά. Έτσι, ενώ υπήρξε προσπάθεια για παράκαμψη του προβλήματος της SSL πιστοποίησης, αποφασίστηκε πώς η εντολή `iframe` είναι ζωτικής σημασίας για τη λειτουργικότητα της εργασίας και θα πρέπει να παραμείνει δυστυχώς με HTTP σύνδεση κάτι το οποίο όμως δεν επηρεάζει κάπου τη λειτουργικότητα της.

4.4.2.3 ΕΠΙΚΟΙΝΩΝΙΑ SECOND LIFE ΠΡΟΣ PHP

Για την αποστολή δεδομένων από το Second Life προς τη σελίδα PHP, η σελίδα PHP ελέγχει αν η μεταβλητή `test`, υπάρχει στην κλήση της διεύθυνσης της με την δομή ελέγχου `isset()` η οποία είναι λογικού τύπου (BOOLEAN).

Επίσης, για τη δημιουργία του αρχείου κειμένου `TEST.txt` χρησιμοποιείται η συνάρτηση `file_put_contents(filename, data, mode, context)`. Σε αυτή τα πεδία `filename` και `data` είναι υποχρεωτικά. Αν δεν υπάρχει αρχείο με το όνομα `filename`, η συνάρτηση θα το δημιουργήσει. Η συγκεκριμένη συνάρτηση απαιτεί τη χρήση έκδοσης PHP 5.0 ή μεταγενέστερη. Η επιστροφή της

συγκεκριμένης συνάρτησης είναι ο αριθμός των byte που ενέγραψε στο αρχείο, σε περίπτωση επιτυχής εγγραφής ή FLASE, σε περίπτωση αποτυχίας[18].

4.4.2.4 ΧΡΗΣΗ AJAX

Στη συγκεκριμένη εργασία, χρησιμοποιείται η χρήση AJAX εντός κώδικα JavaScript που βρίσκεται εμφωλευμένος στη σελίδα HTML.

Αρχικά, η χρήση AJAX λειτουργεί μόνο για επικοινωνία εντός του ίδιου ονόματος Domain, για λόγους ασφαλείας. Όλοι οι σύγχρονοι Web Browser υποστηρίζουν τη χρήση του (Chrome, Firefox, IE7+, Edge, Safari, Opera) [19].

Για την επικοινωνία, χρησιμοποιείται HTTP επικοινωνία με μέθοδο «GET». Η κατάσταση της HTTP επικοινωνίας συνήθως είναι «OK» (status code 200).

ΣΥΝΟΨΗ ΚΕΦΑΛΑΙΟΥ

Η μεταφορά δεδομένων του χρήστη από και προς το Second Life κάνει χρήση πολλαπλών τεχνικών, γλωσσών προγραμματισμού και διαφορετικών πλατφόρμων οι οποίες στο μεγαλύτερο βαθμό επικοινωνούν αυτόματα.

Εντός του Second Life η μεταφορά των δεδομένων γίνεται ανάμεσα στα διαφορετικά αντικείμενα που απαρτίζουν το παιχνίδι Simon Says ενώ χρησιμοποιούνται 3 διαφορετικά κανάλια επικοινωνίας του Second Life για την επίτευξη επικοινωνίας.

Για την επικοινωνία με τον Web Server χρησιμοποιήθηκε επικοινωνία HTTP μέσω URL που παράγεται από τον Server του Second Life, ενώ η τελική διεύθυνση διέπεται με τη χρήση ενός QR κωδικού.

Στον Web Server, για να αποσταλούν δεδομένα στο Second Life, δημιουργείται νέο παράθυρο περιήγησης, στη σελίδα HTML, για κάθε διαφορετική επιλογή του χρήστη, ενώ το παράθυρο αυτό μένει αόρατο με την εγκατάσταση του σε μια περιοχή iframe. Προς τον Web Server, τα δεδομένα διαχειρίζονται από μια σελίδα PHP η οποία τα εγγράφει σε ένα αρχείο κειμένου TXT.

Η ανάγνωση δεδομένων στην HTML σελίδα από το αρχείο κειμένου επιτυγχάνεται με τη χρησιμοποίηση AJAX πράγμα το οποίο εξασφαλίζει τη μη ανανέωση της σελίδας HTML κάθε φορά που πρέπει να αναγνωστούν νέα δεδομένα.

Καθ' όλη τη μεταφορά δεδομένων σε κάθε τομέα της εργασίας, υπάρχουν περιορισμοί μεγέθους δεδομένων, κλήσης συναρτήσεων και απελευθέρωσης διευθύνσεων, οι οποίες στο μεγαλύτερο μέρος

τους δεν επηρεάζουν τη λειτουργικότητα της εργασίας καθώς κάνει χρήση μικρού μεγέθους δεδομένων. Ταυτόχρονα, το κυριότερο πρωτόκολλο που ισχύει κατά την επικοινωνία και την μετάβαση των δεδομένων, είναι το πρωτόκολλο HTTP που χρησιμοποιείται εκτενώς για την κλήση (request) και απάντηση (response) με δεδομένα ανάμεσα στους φορείς.