Ατομική Διπλωματική Εργασία

ΑΝΑΠΤΥΞΗ ΠΛΑΤΦΟΡΜΑΣ ΓΙΑ ΥΛΟΠΟΙΗΣΗ ΣΟΒΑΡΩΝ ΠΑΙΧΝΙΔΙΩΝ ΣΤΗΝ ΕΙΚΟΝΙΚΗ ΤΗΛΕΪΑΤΡΙΚΗ: ΕΦΑΡΜΟΦΗ ΣΤΟ ΗΛΕΚΤΡΟΚΑΡΔΙΟΓΡΑΦΗΜΑ

Χάρης Μαραγκός

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Δεκέμβριος 2012

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Ανάπτυξη πλατφόρμας για υλοποίηση σοβαρών παιχνιδιών στην εικονική τηλεϊατρική: εφαρμογή στο ηλεκτροκαρδιογράφημα

Χάρης Μαραγκός

Επιβλέπων Καθηγητής
Δρ. Κωνσταντίνος Παττίχη

Η Ατομική Διπλωματική Εργασία υποβλήθηκε προς μερική εκπλήρωση των απαιτήσεων απόκτησης του πτυχίου Πληροφορικής του Τμήματος Πληροφορικής του Πανεπιστημίου Κύπρου

Δεκέμβριος 2012

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον Δρ. Κωνσταντίνο Παττίχη που με επέλεξε γι' αυτή τη διπλωματική εργασία και για την επίβλεψη του καθ' όλη τη διάρκεια του έτους. Θα ήθελα επίσης να ευχαριστήσω τον Δρ. Άθω Αντωνιάδη για την πολύτιμη του συνεισφορά και καθοδήγηση, χωρίς την οποία αυτή η εργασία δεν θα ήταν δυνατό να γίνει. Ο Δρ. Άθως Αντωνιάδης είχε προτείνει την ανάπτυξη ενός Σοβαρού Παιχνιδιού το οποίο θα είναι βασισμένο σε ένα σύστημα τηλεϊατρικής. Το σύστημα αυτό αναπτύχθηκε υπό την επίβλεψη του Δρ. Ευθύβουλου Κυριάκου, τον οποίο θα ήθελα επίσης να ευχαριστήσω. Ακολούθως, θα ήθελα να ευχαριστήσω την Δρ. Ιόλη Νικολαΐδου, η οποία με τη συνεισφορά της βελτίωσε την εκπαιδευτική αξία του όλου έργου και βοήθησε την προώθηση του. Τέλος, θα ήθελα να ευχαριστήσω την Δρ. Ελένη Δάφλη και τον Δρ Παναγιώτη Μπαμίδη, οι οποίοι μας βοήθησαν ιδιαίτερα με την ορθότητα του ιατρικού κομματιού του έργου και την ανάπτυξη των δειγματικών σεναρίων.

Περίληψη

Η διπλωματική εργασία αυτή αναλύει την έννοια των Σοβαρών Παιχνιδιών και επεξηγεί πώς τα παιχνίδια αυτά μπορούν να χρησιμοποιηθούν για διαπαιδαγωγικούς σκοπούς σε διάφορους τομείς και συγκεκριμένα όσον αφορά τον ιατρικό τομέα. Ακολουθεί η ανάλυση των απαιτήσεων του Σοβαρού Παιχνιδιού Virtual Telemedicine και επεξηγείται πως αυτό είναι Σοβαρό Παιχνίδι. Αργότερα αναφέρεται πως μπορεί να γίνει η χρήση του παιχνιδιού από έναν απλό χρήστη.

Ακολουθεί μια λεπτομερής ανάλυση της γλώσσας σεναρίων Scribulance, η οποία μπορεί να χρησιμοποιηθεί για τη συγγραφή νέων σεναρίων. Ορίζεται πλήρως η γραμματική της και η μέθοδος με την οποία ένα σενάριο μεταφέρεται από ευανάγνωστο κώδικα σε εκτελέσιμο σενάριο που εμφανίζεται στην οθόνη σαν το τρέχει ο χρήστης. Περιγράφονται δηλαδή όλα τα ενδιάμεσα στάδια που ακολουθούνται για τη μετατροπή του κώδικα σε καθαρές εντολές που εκτελούνται σε πραγματικό χρόνο, καθώς και οι κρυφές προς τον χρήστη ενέργειες που τρέχουν στο παρασκήνιο ώστε να καταστήσουν την εκτέλεση του σεναρίου δυνατή. Η ανάλυση αυτή τελειώνει εξηγώντας πώς η γλώσσα αυτή, παρόλο που γράφτηκε για το Virtual Telemedicine, μπορεί να προσαρμοστεί ανάλογα, χωρίς να γίνει αλλαγή σ' αυτήν, σε οποιοδήποτε έργο που θα μπορούσε να τη χρησιμοποιήσει.

Τέλος, γίνεται μια Αξιολόγηση του παιχνιδιού αυτού στην οποία συμπεριλαμβάνεται και το ερωτηματολόγιο που στάλθηκε σε περίπου 20 γιατρούς που βρίσκονται σε διάφορα μέρη της Ελλάδας. Καταλήγουμε στα συμπεράσματα που φθάσαμε μ' αυτή την εργασία, επεξηγώντας πώς ικανοποιήσαμε τις απατήσεις της ανάλυσης του συστήματος μας και τελειώνουμε με μια προεπισκόπηση στους επόμενους στόχους και τη μελλοντική εργασία που ακολουθεί.

Περιεχόμενα

Κεφάλαιο 1

Εισαγωγή

1.1 Γενικά

Τα συστήματα τηλεϊατρικής είναι απαραίτητα σε περιπτώσεις όπου ο γιατρός βρίσκεται σε απόσταση από τον ασθενή, διότι για πολλούς λόγους ένας ασθενής μπορεί να χρειάζεται άμεση φροντίδα χωρίς να είναι εφικτό να γίνει η μεταφορά του σε νοσοκομείο. Τέτοιο σύστημα είναι και αυτό που αναπτύχθηκε υπό την επίβλεψη του Δρ Ευθύβουλου Κυριάκου και άλλους [1], το οποίο μπορεί να χρησιμοποιηθεί για την περίθαλψη ενός ασθενή καθώς βρίσκεται στο δρόμο προς το νοσοκομείο.

Τα Σοβαρά Παιχνίδια είναι μια ειδική κατηγορία παιχνιδιών, η οποία ασχολείται με τη δημιουργία παιχνιδιών τα οποία έχουν εκπαιδευτική ή ενημερωτική αξία. Τέτοιο παιχνίδι είναι και το Virtual Telemedicine, το οποίο στοχεύει ειδικά την εκπαίδευση των ιατρών που ειδικεύονται σε καρδιαγγειακές παθήσεις και θέλουν να ασχοληθούν με την τηλεϊατρική.

Το παιχνίδι αυτό έχει ήδη δημοσιευτεί σε δύο συνέδρια αξιολόγησης από ομότιμους (peer review conference), [2] και [3], και έχει προσκληθεί να συμπεριληφθεί ως δημοσίευση στο Journal of Medical Internet Research. Οι δύο δημοσιεύσεις αυτές συμπεριλαμβάνονται και ως παραρτήματα Β και Γ.

1.2 Στόχος και Παρακίνηση

Αρχικός μας στόχος ήταν η ανάπτυξη ενός σοβαρού παιχνιδιού το οποίο προσομοιώνει τη χρήση ενός συγκεκριμένου συστήματος τηλεϊατρικής. Το σύστημα αυτό είναι κατάλληλο για τη βελτίωση της επικοινωνίας μεταξύ του νοσοκόμου, ο οποίος βρίσκεται μαζί με τον ασθενή καθοδόν για το νοσοκομείο, και του ιατρού, ο οποίος βρίσκεται στο νοσοκομείο [4]. Ο γιατρός μπορεί να βλέπει μέσω βίντεο τον ασθενή και να λαμβάνει διάφορες πληροφορίες για τον ασθενή όπως τα καρδιογραφήματα του ή την πίεση του. Ταυτόχρονα μπορεί να δίνει οδηγίες στον νοσοκόμο και να λαμβάνει σχόλια απ' αυτόν. Η αναγκαιότητα τέτοιου συστήματος είναι δεδομένη, διότι η απόσταση ενός ασθενή από το νοσοκομείο μπορεί να είναι μεγάλη [1] και έτσι ο χρόνος μεταφοράς του προς το νοσοκομείο μπορεί να αποβεί μοιραίος αν δεν τύχει της σωστής περίθαλψης.

Το παιχνίδι πρέπει επίσης να ανταποκρίνεται πλήρως στις εκπαιδευτικές ανάγκες που θέσαμε πιο πάνω. Μεγάλη ανάγκη είναι η ιατρική ορθότητα του παιχνιδιού, λόγω του ότι ανταποκρίνεται σε πραγματικούς γιατρούς και σε σοβαρές καταστάσεις. Οποιαδήποτε λανθασμένη πληροφορία πρόκειται να στοιχίσει, αφού μπορεί να οδηγήσει τον γιατρό στο να πάρει λανθασμένη απόφαση η οποία να έχει αρνητικές επιπτώσεις στην υγεία του ασθενή.

Τέλος, σημαντικό είναι το παιχνίδι να μπορεί να επεκταθεί και να δεχθεί συνεισφορά από επαγγελματίες γιατρούς οι οποίοι μπορούν να προσθέσουν δικά τους σενάρια που θα καλύπτουν νέα περιστατικά. Αυτός ο στόχος απαιτεί να είναι εύκολος και κατανοητός από άτομα χωρίς προχωρημένη γνώση στον τομέα της Πληροφορικής.

1.3 Σύντομη Περιγραφή

Αυτή η διπλωματική εργασία αρχικά περιγράφει την ανάγκη για δημιουργία του σοβαρού παιχνιδιού Virtual Telemedicine, το οποίο είναι βασισμένο στο σύστημα τηλεϊατρικής που αναφέραμε [1]. Εξηγεί το πώς η χρήση του μπορεί να συνεισφέρει στην εκπαίδευση του ιατρικού προσωπικού που ειδικεύεται σε καρδιοαγγειακές παθήσεις, καθώς και το πώς μπορεί να επεκταθεί από διάφορους γιατρούς που θέλουν να συνεισφέρουν,

βοηθώντας τη μεταφορά γνώσης. Αυτό γίνεται με συγγραφή σεναρίων στη γλώσσα Scribulance, η οποία αναπτύχθηκε ειδικά για τις ανάγκες του παιχνιδιού.

Περιγράφονται επίσης λεπτομέρειες υλοποίησης του παιχνιδιού αυτού και τη μεθοδολογία που ακολουθήθηκε, δίνοντας ιδιαίτερη έμφαση στο πώς ακριβώς εκτελούνται τα σενάρια στο παρασκήνιο. Δίνονται λεπτομέρειες για το πώς γλώσσα σεναρίων Scribulance είναι επεκτάσιμη και έχει την ικανότητα να χρησιμοποιηθεί και για άλλες εφαρμογές. Αναφέρονται επίσης και δυσκολίες που αντιμετωπίσθηκαν στη δημιουργία μιας βιβλιοθήκης που περιέχει καρδιογραφήματα, τα οποία μπορούν να χρησιμοποιηθούν από τους σεναριογράφους για τα σενάρια τους, και πώς τα προβλήματα αυτά επιλύθηκαν.

Κεφάλαιο 2

Εισαγωγή στα Σοβαρά Παιχνίδια

2.1 Εισαγωγή στα Βιντεοπαιχνίδια

Ένα παιχνίδι είναι ένα σύστημα στο οποίο:

1) Παίχτες προσπαθούν να ξεπεράσουν ένα τεχνητό εμπόδιο
2) Ορίζεται από ένα σύνολο από κανόνες
3) Καταλήγει σε ένα μετρίσιμο αποτέλεσμα

Ο πιο πάνω ορισμός [5] είναι αρκετός στο να καθορίσει τί ακριβώς αποτελεί ένα παιχνίδι. Ένα σύστημα δεν είναι παιχνίδι εάν του λείπει ένα συστατικό από τα πιο πάνω. Ακολουθεί μια εις βάθος ανάλυση για κάθε ένα από τα τρία αυτά στοιχεία:

1) **Ένα τεχνητό εμπόδιο**

   Το ξεπέρασμα του τεχνητού εμποδίου λέγεται και στόχος του παιχνιδιού. Είναι τεχνητό διότι ο παίκτης θα μπορούσε, αν ήθελε, να το αγνοήσει – που στην προκειμένη περίπτωση το παιχνίδι διακόπτεται.

   Το εμπόδιο αυτό θα μπορούσε να είναι και μια διαμάχη μεταξύ των παικτών όταν οι παίκτες είναι περισσότεροι από ένας. Μια διαμάχη υπάρχει όταν οι στόχοι των παικτών αντικρούονται, επομένως οι παίκτες προσπαθούν να εκπληρώσουν το στόχο τους ενώ παράλληλα προσπαθούν να αποτρέψουν τους άλλους παίκτες – αντιπάλους από την ολοκλήρωση του δικού τους στόχου.

Μια διαμάχη, ή σύγκρουση, θα μπορούσε να είναι ένας στόχος ο οποίος είναι κοινός μεταξύ των παικτών αλλά δεν μπορούν να μοιραστούν. Για παράδειγμα, σε μια κούρσα όλοι οι παίκτες θέλουν να τερματίσουν πρώτοι. Δεν είναι αναγκαίο όμως να υπάρχει κάποιο «μήλο της έριδος» εφόσον οι στόχοι των παικτών

2) Ένα σύνολο από κανόνες

Οι κανόνες καθορίζουν πώς ακριβώς παίζεται ένα παιχνίδι. Οι κανόνες είναι σημαντικοί διότι τα εμπόδια είναι τεχνητά. Χωρίς κανόνες δεν υπάρχει κάποια δύναμη η οποία μπορεί να περιορίσει τους παίκτες με το εμπόδιο που τους επιβάλλεται. Για παράδειγμα, ο στόχος στο σκάκι είναι για κάθε παίκτη να κατακτήσει τον βασιλιά του αντιπάλου. Εμπόδιο είναι ο άλλος παίκτης και η στρατιά του από πιόνια. Ή δύναμη που περιορίζει ένα παίκτης από το να τεντώσει το χέρι του και να αρπάξει τον βασιλιά του αντιπάλου και έτσι να τελειώσει το παιχνίδι σε δευτερόλεπτα, είναι οι κανόνες.

Οι κανόνες συνήθως υπάρχουν και για να κρατούν κάποια ισορροπία στο παιχνίδι. Με τον όρο ισορροπημένο παιχνίδι εννοούμε ένα παιχνίδι στο οποίο κανένας παίκτης δεν έχει κάποιο άδικο πλεονέκτημα.

3) Ένα μετρήσιμο αποτέλεσμα

Ένα παιχνίδι δεν είναι ολοκληρωμένο εάν δεν υπάρχει κάποιο είδος βαθμολογίας. Το αποτέλεσμα ενός παιχνιδιού είναι αυτό που δίνει το κίνητρο στους παίκτες να καταβάλουν μια προσπάθεια στην εκπλήρωση του στόχου τους. Στην πιο γενική μορφή το αποτέλεσμα ταξινομεί τους παίκτες από τον καλύτερο στον χειρότερο.

Κάποιες φορές είναι αρκετό να ξεχωρίσουμε μεταξύ νικητή και ηττημένου, όπως στο τάβλι ή τη ναυμαχία. Άλλες φορές, όπως στο σκάκι και το ποδόσφαιρο, υπάρχει και το ενδεχόμενο της ισοπαλίας. Άλλα παιχνίδια μπορούν να έχουν πολλούς νικητές και πολλούς χαμένους.

Σε κάποια παιχνίδια όπως το φλίπερ δεν υπάρχει η έννοια του νικητή. Όλοι οι παίκτες παίζουν μέχρι να χάσουν. Έτσι, ο στόχος του κάθε παίκτη είναι να πάρει όσο το δυνατό πιο ψηλή βαθμολογία. Οι παίκτες τότε ταξινομούνται ανάλογα με τη βαθμολογία τους, χωρίς να υπάρχει όμως η διάκριση του νικητή.

Το παίξιμο ενός παιχνιδιού έχει συνήθως ως στόχο ένα ή περισσότερα από τα ακόλουθα:

- Την ψυχαγωγία
- Την εκπαίδευση
- Την καλλιέργεια κοινωνικών σχέσεων
- Την απόκτηση κάποιου βραβείου ή επάθλου
- Την ικανοποίηση του ανταγωνιστικού πνεύματος και της προβολής ικανοτήτων, τα οποία είναι έμφυτα στον άνθρωπο

Ένα Ηλεκτρονικό Παιχνίδι είναι ένα παιχνίδι το οποίο παίζεται με τη βοήθεια μιας ηλεκτρονικής συσκευής η οποία λειτουργεί και ως διαπροσωπικά μεταξύ του παίκτη και του παιχνιδιού. Παραδείγματα Ηλεκτρονικών Παιχνιδιών είναι το φλιπεράκι, τα παιχνίδια τύχης στα καζίνο και παρόμοιες μηχανές σε άλλα κέντρα αναψυχής [6].

Τα Βιντεοπαιχνίδια είναι Ηλεκτρονικά Παιχνίδια στα οποία οι παίκτες λαμβάνουν ανταπόκριση από μια οθόνη. Είναι μια πιο ειδική κατηγορία των Ηλεκτρονικών Παιχνιδιών και οι δύο έννοιες δεν είναι ταυτόσημες, παρόλο που πολλές φορές λανθασμένα θεωρούνται πως είναι.

Στην παραδοσιακή τους μορφή, τα Βιντεοπαιχνίδια παίζονται με κάποιου είδους χειριστήριο, το οποίο στέλνει εντολές στο παιχνίδι. Οι παίκτες συνήθως ελέγχουν την κίνηση ενός ειδώλου τους που βρίσκεται μέσα στον κόσμο του παιχνιδιού και εκτελούν ειδικές ενέργειες με τη χρήση κουμπιών. Ο έλεγχος ενός παιχνιδιού μπορεί να πράττεται μέσω ενός ή περισσότερων από τα ακόλουθα:

- Μοχλός κίνησης ή κουμπιά κατεύθυνσης
- Ειδικά κουμπιά ενεργειών
- Πληκτρολόγιο
- Ποντίκι

- Οθόνες επαφής
- Συσκευές αναγνώρισης κίνησης
- Μικρόφωνα
- Τηλεχειριστήρια
- Επιταχυνσιόμετρα
- Ειδικά κατασκευασμένες μηχανικές συσκευές όπως τιμόνια
- Άλλοι τρόποι

Υπάρχουν πολλές κατηγορίες και κατηγοριοποιήσεις Βιντεοπαιχνιδιών. Μια κατηγοριοποίηση, η οποία μπορεί να συμπεριλάβει όλα τα βιντεοπαιχνίδια, θα μπορούσε να είναι η εξής:

- Δράσης
- Περιπέτειας
- Στρατηγικής / Διαχείρισης Πόρων
- Ρόλων
- Σπαζοκεφαλιάς
- Προσομοίωσης
- Αθλημάτων
- **Εκπαιδευτικά / Σοβαρά Παιχνίδια**

Αξίζει να σημειωθεί πως οποιαδήποτε κατηγοριοποίηση Βιντεοπαιχνιδιών, όπως και η πιο πάνω, είναι υποκειμενική. Τέλος, ένα Βιντεοπαιχνίδι ενδέχεται να ανήκει σε περισσότερες από μία κατηγορίες.

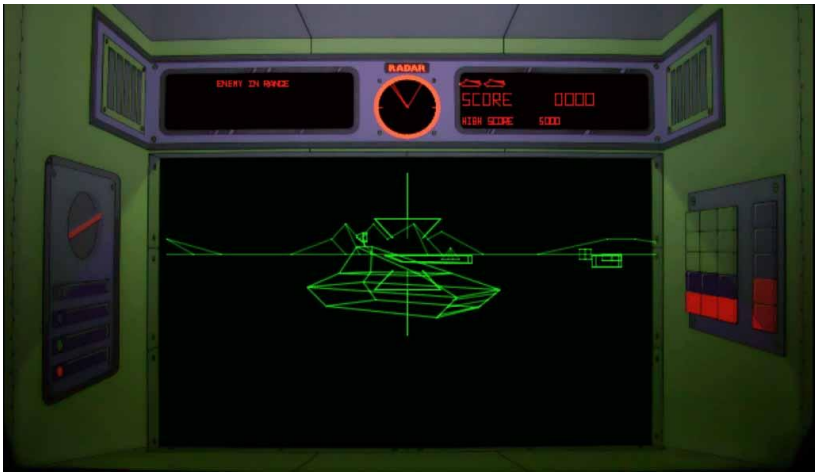
2.2 Η Έννοια των Σοβαρών Παιχνιδιών

Ένα Σοβαρό Παιχνίδι είναι μια ειδική κατηγορία παιχνιδιών η οποία ασχολείται με παιχνίδια που έχουν ως στόχο την εκπαίδευση και όχι τη ψυχαγωγία, χωρίς όμως να την αποκλείουμε. Ορίζονται ως "παιχνίδια με εκπαιδευτικό σκοπό... που εμπλέκουν το χρήστη... έχουν παιδαγωγικό υπόβαθρο όπου η μάθηση μπορεί να είναι άμεση ή έμμεση" [7].

Τα σημερινά ψηφιακά παιχνίδια είναι χαρακτηριστικά με μεγάλη διάρκεια, δύσκολα και πολύπλοκα. Για να «νικήσουν» με επιτυχία ένα παιχνίδι και ολοκληρώσουν τον μακροπρόθεσμο στόχο του παιχνιδιού, οι παίκτες συνήθως πρέπει να αναπτύξουν μια ποικιλία νέων στρατηγικών και δεξιοτήτων. Αυτό έχει οδηγήσει μερικούς ακαδημαϊκούς στην αναγνώριση των βιντεοπαιχνιδιών ως «οριοθετημένο πρόγραμμα εκμάθησης». [8]

Οι εφαρμογές των σοβαρών παιχνιδιών είναι πολλές και ασχολούνται με διάφορους τομείς, όπως τη σχολική παιδεία [9], την ιατρική και τον υγειονομικό τομέα [3], τον στρατό [10][11], τις επιχειρήσεις [12], τη διαφήμιση [13].

Τα Σοβαρά Παιχνίδια δεν είναι πρόσφατη ιδέα. Το πρώτο σοβαρό παιχνίδι θεωρείται το Army Battlezone το 1980, μια ειδική έκδοση του Battlezone (Σχήμα 2.1) που προορίζονταν για τον αμερικάνικο στρατό [14].



Σχήμα 2.1: Το παιχνίδι Battlezone [15]

Ένα από τα πιο γνωστά και εμπορικά επιτυχή σοβαρά παιχνίδια είναι το Microsoft Simulator [16], το οποίο είναι μια από τις πιο μακρόχρονες σειρές παιχνιδιών, με σύνολο 12 τίτλους από το 1982 μέχρι το 2006. Σ' αυτό το παιχνίδι ο χρήστης λαμβάνει τον ρόλο ενός πιλότου (Σχήμα 2.2) και έχει ως αποστολή του την επιτυχής απογείωση, πτήση και προσγείωση ενός αεροπλάνου. Υποστηρίζει διάφορα μοντέλα αεροπλάνων και τη δυνατότητα προσθήκης επιπλέον μοντέλα από τρίτους.

Σχήμα 2.2 Το πιλοτήριο ενός αεροπλάνου μέσα στο παιχνίδι Microsoft Flight Simulator X [17]

Το παιχνίδι αυτό χρησιμοποιείται για την εκπαίδευση μαθητευόμενων πιλότων [18] αλλά και από έμπειρους πιλότους που επιθυμούν να ανανεώσουν ή να αυξήσουν τις γνώσεις τους. Σύμφωνα με τον Stefano Oss[19], μπορεί ακόμα να χρησιμοποιηθεί και για άλλους σκοπούς, όπως τη μελέτη της φυσικής ιπτάμενων αντικειμένων.

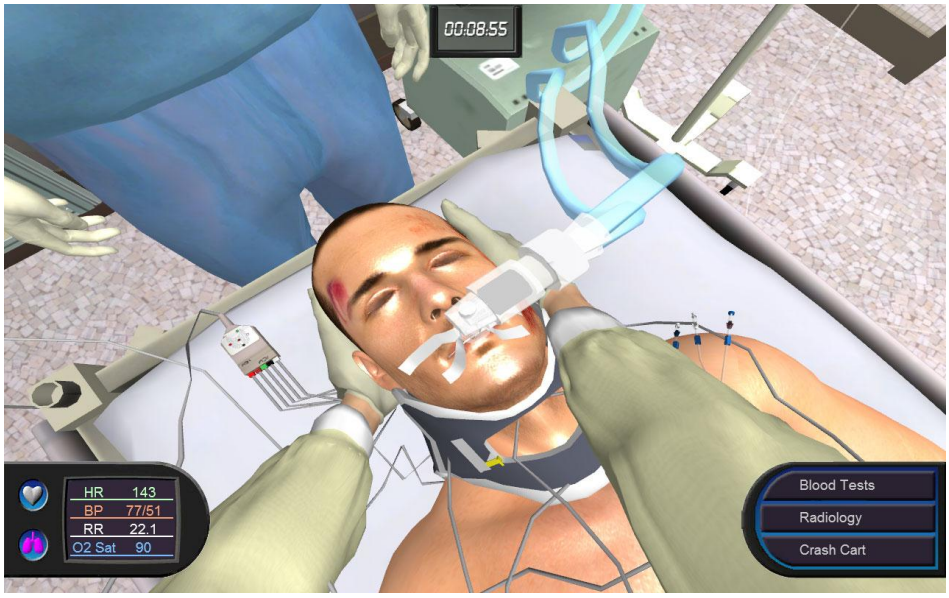
2.3 Τα Σοβαρά Παιχνίδια στην Ιατρική

Η αξιοποίηση των σοβαρών παιχνιδιών για την ιατρική εκπαίδευση είναι μια συνεχώς αναπτυσσόμενη περιοχή. Παιχνίδια που στοχεύουν τους επαγγελματίες υγείας, όπως για παράδειγμα γιατροί και νοσηλευτές, τείνουν να είναι στη βάση τους προσομοιωτές και χρησιμοποιούνται για την εκπαίδευση [7]. Το [20] αναφέρεται στο παράδειγμα της χειρουργικής εκπαίδευσης ως μια εφαρμογή των προσομοιώσεων στον ιατρικό τομέα για την υποστήριξη της κατάρτισης και της μάθησης. Μια άλλη εφαρμογή των προσομοιώσεων στην ιατρική εκπαίδευση αναφέρεται στη χρήση των avatar για σκοπούς εκπαίδευσης, με στόχο να παρακινήσει τους βετεράνους που παρουσιάζουν συμπτώματα του στρες στο να ζητήσουν βοήθεια [21]. Επιπλέον, προηγμένες ιατρικές προσομοιώσεις έχουν χρησιμοποιηθεί για να διευκολύνουν την ενδοσκοπική χειρουργική εκπαίδευση και ως εκ τούτου βελτίωσαν και την ασφάλεια των ασθενών [22].

Σύγχρονες τάσεις στην έρευνα και ανάπτυξη των σοβαρών παιχνιδιών για την υγειονομική περίθαλψη επικεντρώνεται στην επαγγελματική κατάρτιση, τη βελτίωση της απο-

τελεσματικότητας των θεραπειών για τους ασθενείς, καθώς και την προώθηση και ευαισθητοποίηση των σχετικών θεμάτων υγείας σε ένα ευρύτερο κοινό [23]. Το περιβάλλον τέτοιων παιχνιδιών παρέχει ένα ασφαλές και ελεγχόμενο περιβάλλον μέσα στο οποίο οι παίκτες μπορούν να εμπλακούν μαθαίνοντας, είτε πρόκειται για εξερεύνηση νέων κλινικών τεχνικών, που υποβάλλονται σε δραστηριότητες αποκατάστασης, ή να εκτίθενται σε θέματα υγείας.

Ένα παράδειγμα από μία εφαρμογή σοβαρού παιχνιδιού που χρησιμοποιεί ένα εικονικής πραγματικότητας χώρο εκμάθησης, το οποίο αναπτύχθηκε για την εκπαίδευση των επαγγελματιών υγείας σε κλινικές δεξιότητες, είναι το Pulse!! – The Virtual Clinical Learning Lab (Σχήμα 2.3). Σε αυτό το παιχνίδι τα γραφικά αναδημιουργούν ένα ζωντανό, διαδραστικό, εικονικό περιβάλλον εκπαίδευσης στο οποίο πολιτικοί και στρατιωτικοί επαγγελματίες υγείας εξασκούν κλινικές δεξιότητες, ώστε να μπορέσουν να ανταποκριθούν καλύτερα στους τραυματισμούς κατά τη διάρκεια καταστροφικών συμβάντων, όπως εν διάρκεια πολεμικής διαμάχης ή η βιοτρομοκρατίας. Το παιχνίδι έχει σχεδιαστεί για να υποστηρίξει μια σειρά αναγκών που απαιτούνται για την κατάρτιση των νοσηλευτών και επαγγελματιών υγείας [20].

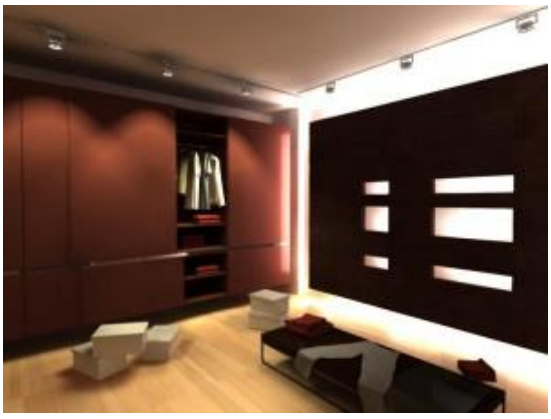


Σχήμα 2.3: Ένας ασθενής δέχεται περίθαλψη από τον χρήστη [24]

Ένα αρκετά ενδιαφέρον παιχνίδι είναι το R.O.G.E.R [25], το οποίο είναι το πρώτο ιατρικό σοβαρό παιχνίδι που χρησιμοποιεί το Kinect της Microsoft για τον χειρισμό του.

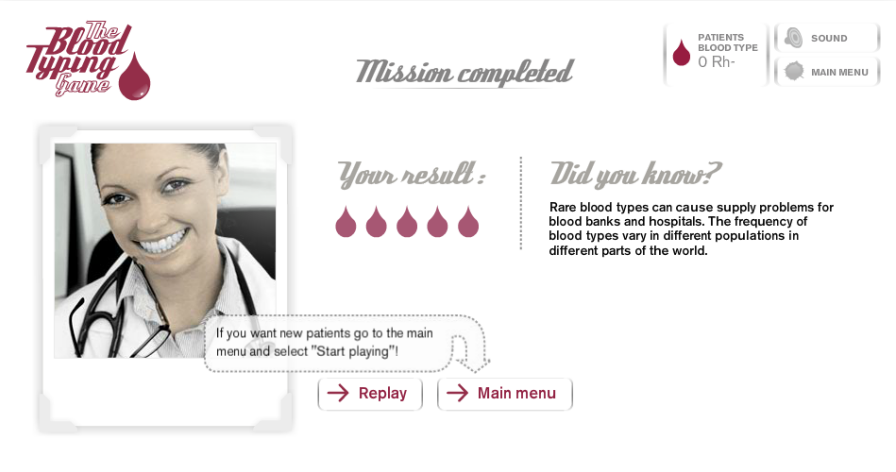
Το παιχνίδι αυτό αφιερώνεται στους ασθενείς που πάσχουν από αποδιοργάνωση και έλλειψη συγκέντρωσης, όπως ασθενείς που πάσχουν από Αλτσχάιμερ ή ασθενείς που έχουν περάσει από εγκεφαλικό επεισόδιο. Το σενάριο αυτού του παιχνιδιού λαμβάνει μέρος μεταξύ τριών δωματίων (υπνοδωμάτιο (Σχήμα 2.4), μπάνιο και καμαρίνι (Σχήμα 2.5)), τα οποία περιέχουν διάφορα αντικείμενα. Ένας ασθενής, ο οποίος πρέπει να φύγει από το σπίτι του για διάφορους προορισμούς, είτε είναι σε ταξίδι είτε σε πόλη, πρέπει να φτιάξει τις αποσκευές που θα χρειαστεί. Ο χρήστης-θεραπευτής τότε παρατηρεί τον ασθενή και ποια αντικείμενα συλλέγει για το ταξίδι του. Από εδώ μπορεί να δει κατά πόσο ο ασθενής είναι αποδιοργανωμένος και πού δυσκολεύεται να αποφασίσει λογικά, έτσι ώστε η θεραπεία του να μπορέσει να εστιαστεί στα σωστά σημεία.

Σχήμα 2.4 Το υπνοδωμάτιο[25]

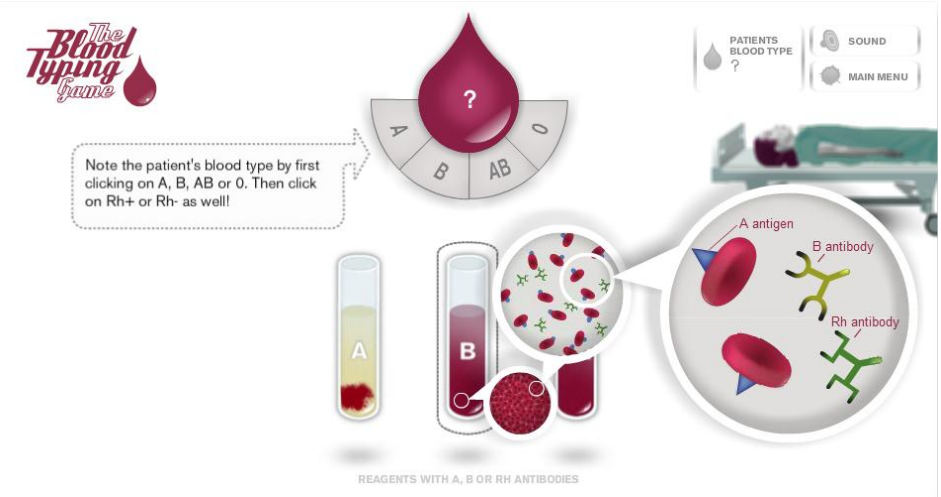


Σχήμα 2.5 Το καμαρίνι [25]

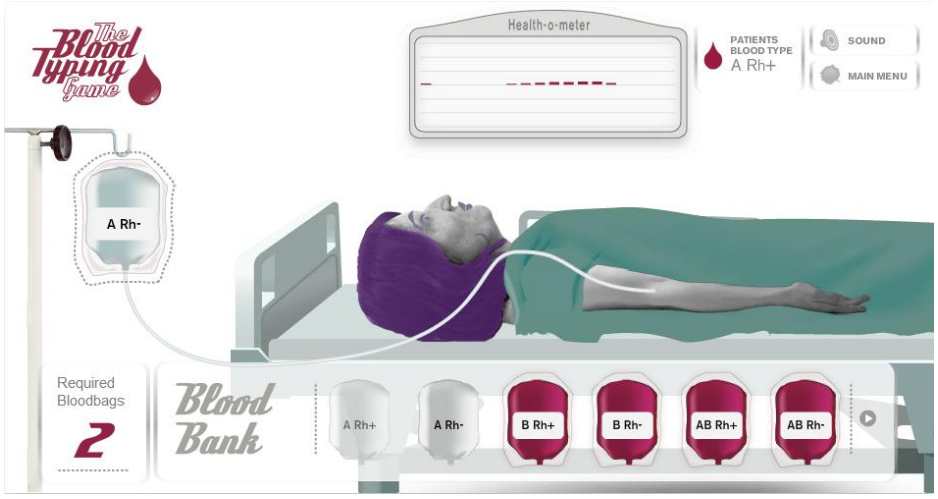
Ένα άλλο παράδειγμα ιατρικού σοβαρού παιχνιδιού είναι το The Blood Typing Game. Αυτό το παιχνίδι χορηγήθηκε από το Ίδρυμα Νόμπελ και είναι βασισμένο στο βραβείο Νόμπελ που δόθηκε για την ανακάλυψη των ανθρώπινων ομάδων αίματος [26]. Το παιχνίδι είναι ασχολείται με την τυποποίηση (Σχήμα 2.7) και μετάγγιση (Σχήμα 2.8) αίματος. Η δουλειά του χρήστη είναι να αποφασίσει σε τι τύπο αίματος ανήκει ο ασθενής, προκειμένου οι μεταγγίσεις αίματος να είναι ασφαλείς. Στο τέλος αξιολογείται με βάση την επίδοση του: Αν δεν κάνει λάθος παίρνει 5 σταγόνες αίματος (Σχήμα 2.6), διαφορετικά παίρνει μια σταγόνα λιγότερη για κάθε λάθος του. Το παιχνίδι μπορεί να παιχτεί σαν σειρά αποστολών, όπου οι παίκτες προσπαθούν να συγκεντρώσουν την υψηλότερη βαθμολογία και να συγκριθούν με αυτή άλλων παικτών [27]. Το παιχνίδι διατίθεται δωρεάν στον σύνδεσμο:
http://www.nobelprize.org/educational/medicine/bloodtypinggame/game/index.html



Σχήμα 2.6 Βαθμολογία με άριστα[28]



Σχήμα 2.7 Τυποποίηση αίματος ενός ασθενή με τύπο αίματος A-[28]



Σχήμα 2.8 Μετάγγιση Αίματος [28]

Κεφάλαιο 3

Απαιτήσεις Συστήματος

3.1 Συγκρότηση Ομάδας Συλλογής Απαιτήσεων

Ως πρώτο βήμα που προηγήθηκε της υλοποίησης του παιχνιδιού ήταν η συγκρότηση μιας ομάδας επιστημόνων οι οποίοι με την πολύτιμη συνεισφορά τους οδήγησαν στον καθορισμό των απαιτήσεων που έπρεπε να ικανοποιηθούν από το παιχνίδι.

Ο Δρ. Άθως Αντωνιάδης μαζί με τον Δρ. Ευθύβουλο Κυριάκου ήταν οι τεχνολογικοί σύμβουλοι όσον αφορά το κομμάτι της τηλεϊατρικής, και η συνεισφορά τους ήταν κυρίως για τη λειτουργικότητα του γραφικού περιβάλλοντος και τη γενική ιδέα του παιχνιδιού.

Η Δρ. Ιόλη Νικολαΐδου ήταν ο εκπαιδευτικός σύμβουλος της περιοχής των Σοβαρών Παιχνιδιών. Η συνεισφορά της οδήγησε άμεσα στη βελτίωση της εκπαιδευτικής πλευράς του παιχνιδιού.

Οι Δρ. Ελένη Δάφλη και Δρ. Παναγιώτης Μπαμίδης ήταν οι γιατροί οι οποίοι μας βοήθησαν με την ορθότητα και επέκταση του ιατρικού κομματιού.

3.2 Απαιτήσεις

Το παιχνίδι πρέπει να ικανοποιεί τις εξής γενικές απαιτήσεις:

- Πρέπει να βασίζεται σε μηχανισμό σεναρίων, τα οποία να μπορούν να προστεθούν ακόμη και μετά την εγκατάσταση του παιχνιδιού. Το κάθε σενάριο είναι ξεχωριστό και η ύπαρξη του δεν επηρεάζει άλλα σενάρια.
- Πρέπει να παρέχεται μηχανισμός συγγραφής σεναρίων, ο οποίος να είναι αρκετά απλός ώστε να μπορεί οποιοσδήποτε να γράψει ένα σενάριο χωρίς να έχει προχωρημένες γνώσεις πληροφορικής.

Το παιχνίδι έπρεπε να ικανοποιεί τις εξής απαιτήσεις όσον αφορά το Γραφικό Περιβάλλον

- Το γραφικό περιβάλλον πρέπει να βασίζεται στη διάταξη του γραφικού περιβάλλοντος ενός συστήματος τηλεϊατρικής [1] (Σχήμα 3.1).
- Ο χρήστης πρέπει να μπορεί να λαμβάνει χρήσιμες πληροφορίες για τον ασθενή: το καρδιογράφημα, τη θερμοκρασία, την πίεση αίματος και την αναπνοή.
- Ο χρήστης πρέπει να μπορεί να δώσει εντολές στο νοσοκόμο και να λάβει παρατηρήσεις από αυτόν.
- Ο χρήστης πρέπει να μπορεί να δει τον ασθενή μέσω κάποιου συστήματος βίντεο.
- Ο χρήστης πρέπει να έχει τη δυνατότητα να βλέπει όλα τα 12 σήματα ενός καρδιογραφήματος.
- Ο χρήστης πρέπει να μπορεί να δει τον χρόνο που έχει περάσει από την αρχή του σεναρίου.

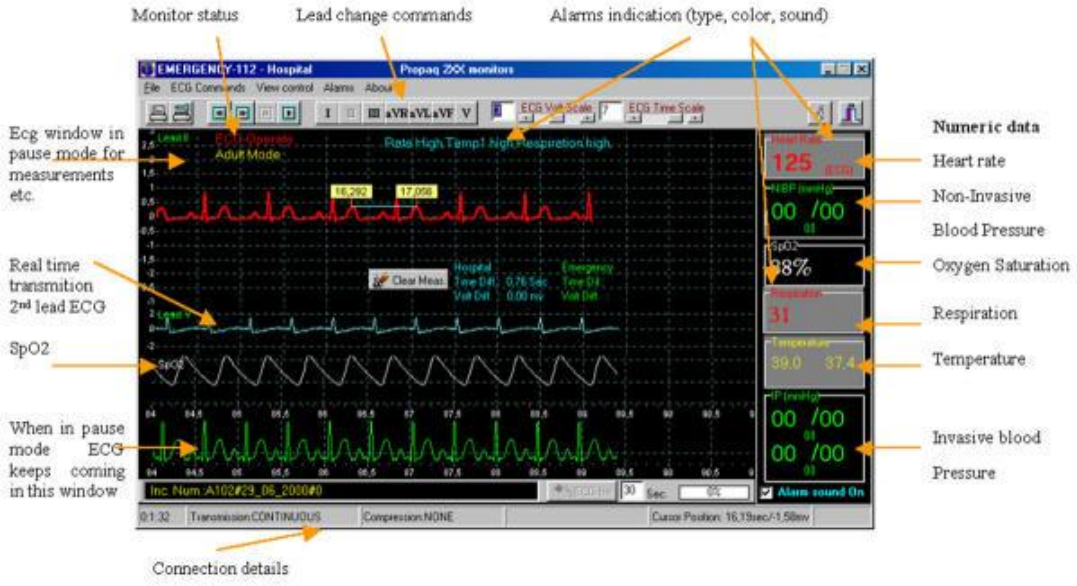
Το παιχνίδι έπρεπε να ικανοποιεί τις εξής απαιτήσεις όσον αφορά το ιατρικό κομμάτι:

- Το παιχνίδι πρέπει να είναι ιατρικά σωστό και να μην θυσιάζεται η ιατρική ορθότητα για χάρη της ψυχαγωγίας.
- Τα σήματα που είναι πιθανό να εμφανιστούν πρέπει να είναι αρκετά για να καλύψουν ένα πολύ ευρύ φάσμα καρδιολογικών παθήσεων.
- Ο χρήστης θα πρέπει να μπορεί να δει το ιστορικό του ασθενή.

Το παιχνίδι έπρεπε να ικανοποιεί τις εξής απαιτήσεις όσον αφορά το εκπαιδευτικό κομμάτι:

- Πρέπει να υπάρχει η δυνατότητα βαθμολογίας της επίδοσης του χρήστη.

- Κάθε πράξη του χρήστη πρέπει να μπορεί να οδηγήσει σε κάποιο αποτέλεσμα, είτε σε μορφή μηνύματος κειμένου, είτε σε αντίδραση του ασθενή.
- Ο εκπαιδευτικός στόχος, ο σκοπός και οι οδηγίες ενός σεναρίου πρέπει να φαίνονται ξεκάθαρα.



Σχήμα 3.1: Το γραφικό περιβάλλον του λογισμικού ενός συστήματος τηλεϊατρική [1]

Κεφάλαιο 4

Σχεδιασμός και Μεθοδολογία

4.1 Διαμοίραση των κομματιών που αποτελούν το παιχνίδι

Το «Virtual Telemedicine» κατασκευάστηκε έχοντας υπ' όψη τη δυνατότητα επέκτασης όσον αφορά τα σενάρια του, ώστε να μπορεί ο καθένας να προσθέτει τα δικά του χωρίς να έχει την ανάγκη να έχει τον πηγαίο κώδικα του παιχνιδιού. Για το σκοπό αυτό το όλο έργο είναι χωρισμένο σε τρία μέρη.

1) **Μεταγλωττιστής**

   Ο Μεταγλωττιστής λαμβάνει ένα σενάριο που είναι γραμμένο στη γλώσσα σεναρίων Scribulance και τη μετατρέπει σε μια ενδιάμεση γλώσσα (Assembulance) που μπορεί πολύ εύκολα μετά να επεξεργαστεί γραμμικά. Έτσι, ο συγγραφέας σεναρίων δεν είναι απαραίτητο να γνωρίζει την ενδιάμεση γλώσσα, η οποία είναι πολύ δύσκολο να γραφτεί από κάποιον άνθρωπο αλλά είναι εύκολο να διαβαστεί από τον υπολογιστή.

   Το κομμάτι αυτό γράφτηκε σε γλώσσα C με τη χρήση του αναγνωριστή συμβόλων Flex και συντακτικού αναλυτή Bison. Μεταγλωττίστηκε σε εκτελέσιμο αρχείο για Windows χρησιμοποιώντας τον MinGNU.

2) **Εκτελεστής Σεναρίων**

   Ο Εκτελεστής Σεναρίων έχει τη δυνατότητα να επεξεργαστεί και να εκτελέσει ένα σενάριο το οποίο είναι γραμμένο στην ενδιάμεση γλώσσα Assembulance.

   Γράφτηκε σε γλώσσα C# και μπορεί να λειτουργήσει ως βιβλιοθήκη για .NET

3) **Παιχνίδι**

   Το παιχνίδι είναι αυτό που βλέπει ο παίκτης. Λειτουργεί ως διαπροσωπεία της επικοινωνίας του με τον Εκτελεστή Σεναρίων, ο οποίος δέχεται εντολές από το παιχνίδι.

   Γράφτηκε σε γλώσσα C# και χρησιμοποιεί το framework του XNA για απεικόνιση των γραφικών μερών.

Καθένα από τα τρία αυτά μέρη είναι αυτόνομο και μπορεί να αντικατασταθεί. Υπάρχει όμως μια συσχέτιση μεταξύ του Μεταγλωττιστή και του Εκτελεστή Σεναρίων, διότι ο Μεταγλωττιστής παράγει είσοδο για τον Εκτελεστή. Αλλά, δεδομένου ότι η ενδιάμεση γλώσσα παραμένει αναλλοίωτη, καθένα απ' αυτά τα δύο κομμάτια μπορεί να ξαναγραφτεί ή να βελτιστοποιηθεί ανεξάρτητα.

Μάλιστα, ο Μεταγλωττιστής και Εκτελεστής σεναρίων είναι γραμμένοι με τέτοιο τρόπο ώστε να είναι επαναχρησιμοποιήσιμοι για εφαρμογές άσχετες με το Virtual Telemedicine. Αυτό θα το δούμε καλύτερα στη συνέχεια.

4.2 Η γλώσσα Σεναρίων «Scribulance»

4.2.1 Στόχος της γλώσσας

Η γλώσσα έχει ως στόχο τη συγγραφή σεναρίων για το παιχνίδι. Παρέχει μηχανισμούς που επιτρέπουν να γίνεται αυτό εύκολα, ενώ αποφεύγει να δώσει περισσότερες δυνατότητες απ' όσες χρειάζεται ο σεναριογράφος.

Ένα σενάριο προγραμματιστικά και σε γενικές γραμμές χωρίζεται σε Καταστάσεις (States), που είναι το βασικό στοιχείο ενός σεναρίου. Κατά τη διάρκεια εκτέλεσης ενός σεναρίου, η εκτέλεση βρίσκεται πάντα σε μία ακριβώς Κατάσταση. Μια Κατάσταση έχει μια συλλογή από Επιλογές (Options) και Γεγονότα (Events).

Οι Επιλογές και τα Γεγονότα συνοδεύονται με μια σειρά από εντολές οι οποίες εκτελούνται όταν ο χρήστης επιλέξει μια τέτοια εντολή ή όταν ένα τέτοιο γεγονός συμβεί. Οι εντολές αυτές μπορούν να αλλάξουν την τρέχουσα κατάσταση, να δείξουν κάποιο μήνυμα στην οθόνη, να εκτελέσουν κάποιες πράξεις, να τερματίσουν το σενάριο και άλλα πολλά.

Η γλώσσα είναι πλήρως επεκτάσιμη από την εφαρμογή που τη χρησιμοποιεί. Ενώ η γραμματική της γλώσσας παραμένει σταθερή, κάθε εφαρμογή που τη χρησιμοποιεί (όπως και το Virtual Telemedicine) μπορεί να ορίσει εντολές ειδικές μόνο γι' αυτή την εφαρμογή.

4.2.2 Γραμματική

Όπως όλες οι γλώσσες, έτσι και η Scribulance ακολουθεί κάποιο γραμματικό δέντρο για να μπορεί να αναγνωρίζει εύκολα και αποδοτικά εάν ένα σενάριο είναι γραμματικά ορθό με βάσει τους νόμους της γλώσσας.

Τα σχήματα από το Σχήμα 4.1 μέχρι το Σχήμα 4.10 περιγράφουν το δέντρο αυτό σε απλουστευμένη μορφή. Οι συμβάσεις που ακολουθούνται είναι:

- Σύμβολα ή λατινικοί χαρακτήρες για *τερματικά σύμβολα* – δηλαδή, αυτά αναγράφονται κυριολεκτικά όπως απαιτεί η γλώσσα.
- Ελληνικοί χαρακτήρες για τους *κανόνες* – όλοι οι κανόνες πρέπει να καταλήγουν σε τερματικά σύμβολα.
- *Προαιρετικοί κανόνες* ή *προαιρετικά τερματικά σύμβολα* περιβάλλονται με **[ ]**
- Στοιχεία που βρίσκονται σε οριζόντια σειρά φανερώνουν *σύζευξη*, δηλαδή πρέπει να παρουσιάζονται όλα τα στοιχεία μιας σύζευξης στη σωστή σειρά για να ισχύει ένας κανόνας

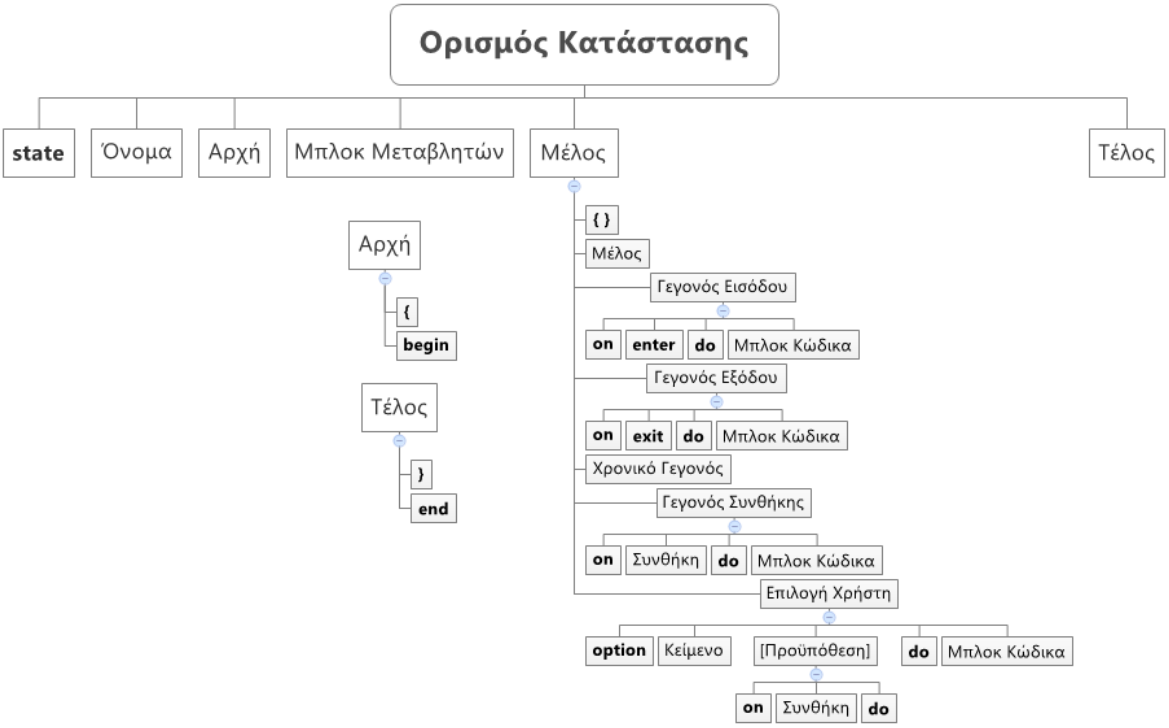
- Στοιχεία που βρίσκονται σε κάθετη σειρά φανερώνουν *διάζευξη*, δηλαδή για να ισχύει ένας κανόνας αρκεί να ισχύει ένα απ' αυτά.
- Εννοείται πως η σύζευξη είναι μεγαλύτερης προτεραιότητας από τη διάζευξη.
- Κάποια τερματικά σύμβολα παρουσιάζονται με *κανονικές εκφράσεις*. Αυτές συμβολίζονται με **regex: <έκφραση>**.

Σχήμα 4.1: Το Σενάριο ορίζει τη ρίζα του γραμματικού δέντρου

Σχήμα 4.2: Το Μπλοκ Μεταβλητών ορίζει μια περιοχή όπου μπορούν να δηλωθούν μεταβλητές

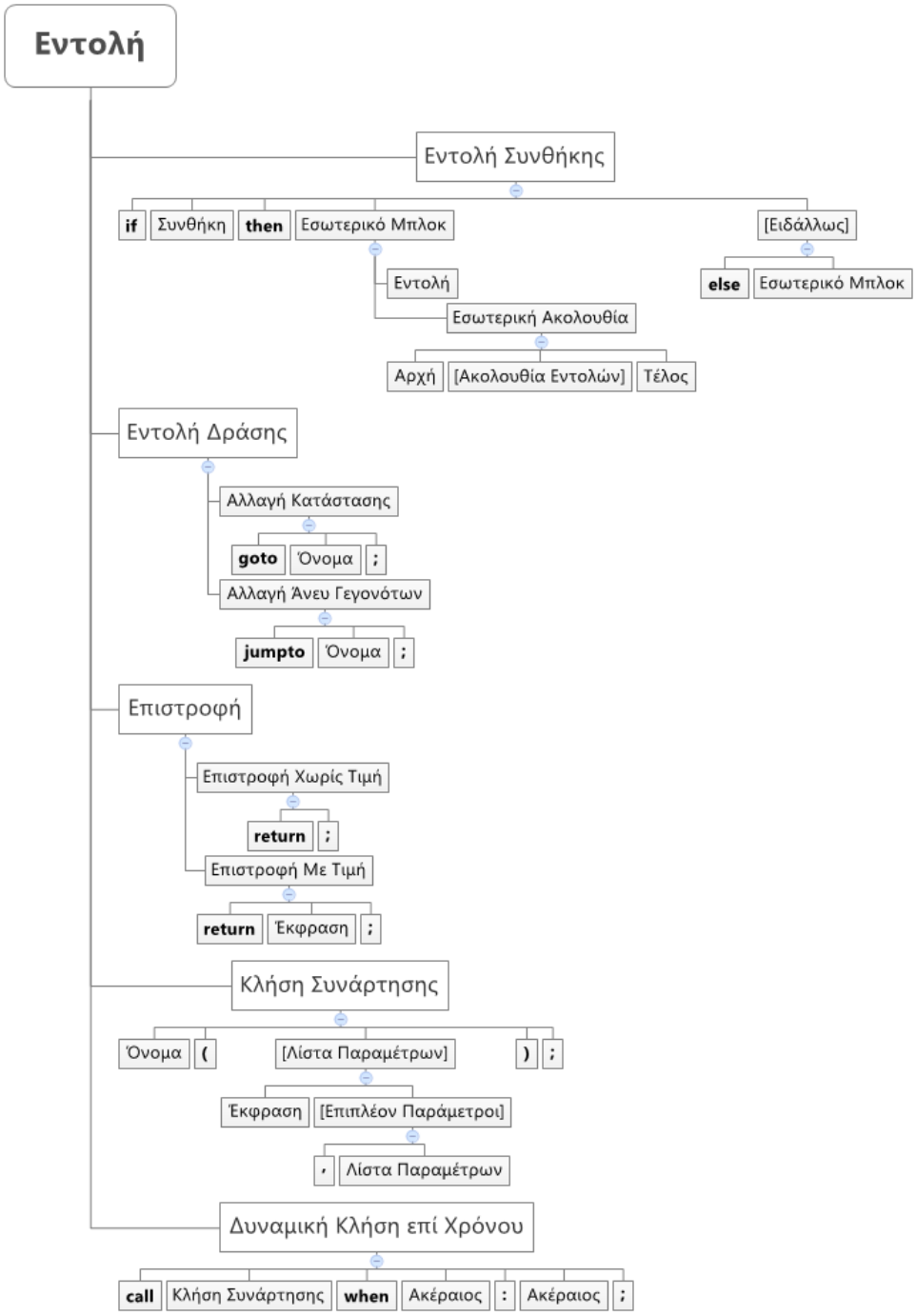
Σχήμα 4.3: Ορισμός Συνάρτησης

Σχήμα 4.4: Χρονικό Γεγονός

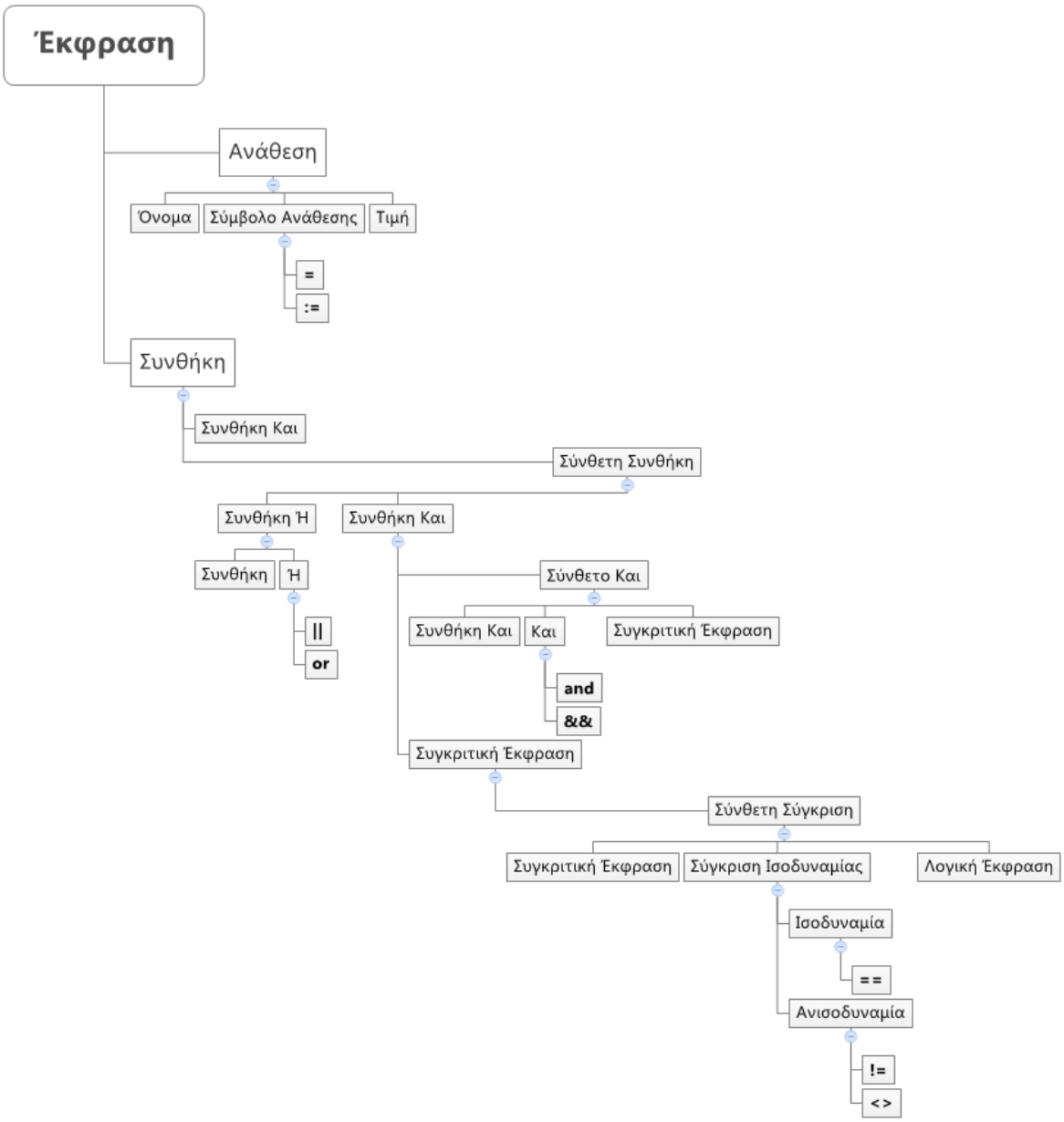


Σχήμα 4.5: Ορισμός Κατάστασης

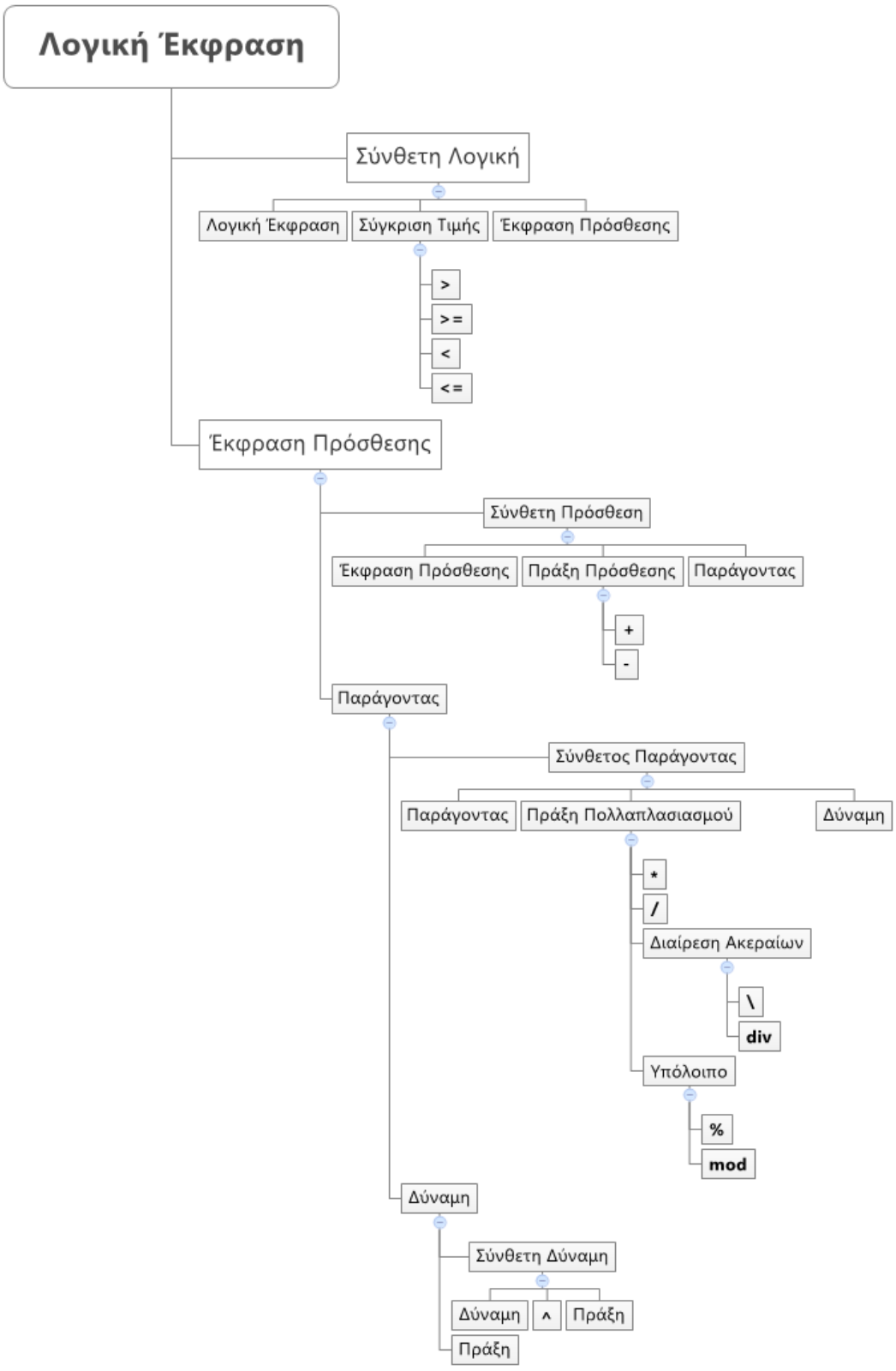
Σχήμα 4.6: Μπλοκ Κώδικα



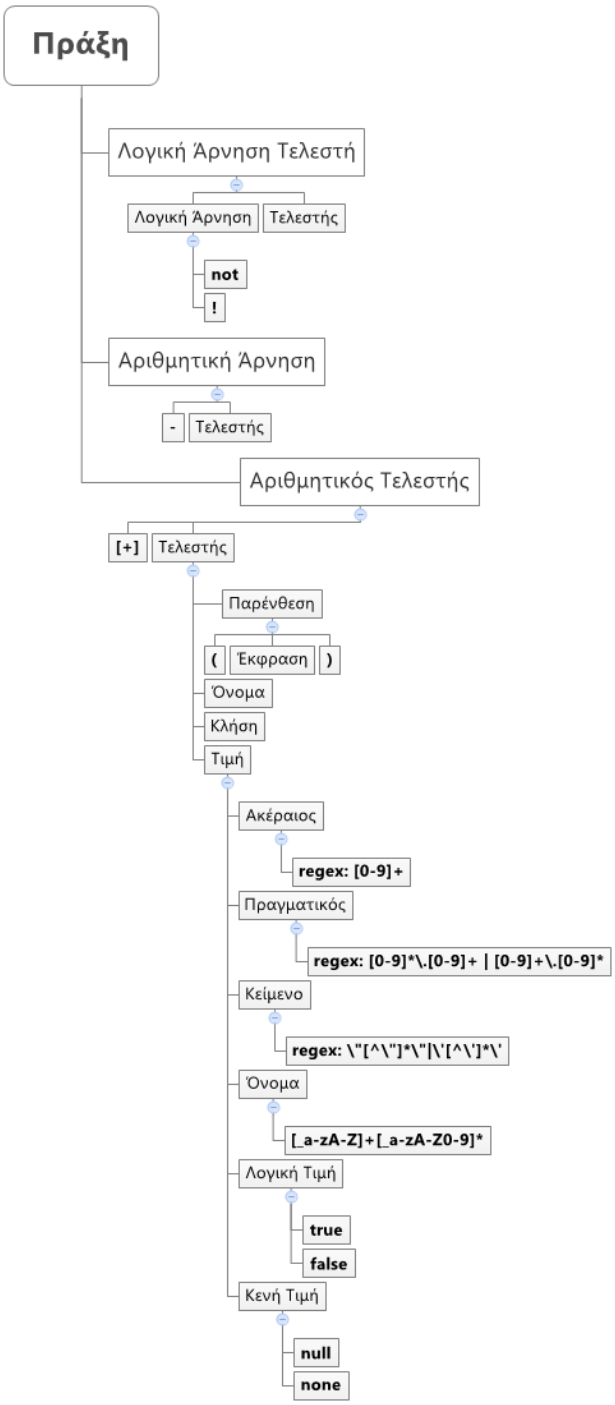
Σχήμα 4.7: Εντολή



Σχήμα 4.8: Μια έκφραση αποτιμείται σε κάποια τιμή



Σχήμα 4.9: Λογική Έκφραση



Σχήμα 4.10: Το πιο χαμηλό επίπεδο του δέντρου γραμματικής.

Ένα σενάριο για να μεταγλωττίσει πρέπει κάθε κομμάτι του να μπορεί να ταιριάξει με τους κανόνες της γραμματικής και τη σειρά τους όπως αναγράφονται στο δέντρο. Αν αυτό δεν ισχύει, τότε ο μεταγλωττιστής θα επιστρέψει μήνυμα λάθους. Αυτό όμως δεν είναι αρκετό, διότι ένα σενάριο πρέπει να υπακούει και σε άλλους κανόνες. Για παρά-

δειγμα, οποιαδήποτε αναφορά γίνει σε μεταβλητή που δεν έχει δηλωθεί θεωρείται λάθος.

Τα τερματικά σύμβολα χωρίζονται με χαρακτήρες κενού. Μπορούν να υπάρχουν ένας ή περισσότεροι χαρακτήρες κενού μεταξύ των τερματικών συμβόλων, αλλά στο τέλος αγνοούνται από τον μεταγλωττιστή.

Για περισσότερες πληροφορίες όσον αφορά τους κανόνες της γλώσσας συμβουλευτείτε το **Παράρτημα Α**.

Ο μεταγλωττιστής χωρίζεται σε τρία μέρη, από τα οποία περνά το σενάριο με αυτή τη σειρά:

1) **Προεπεξεργαστής**
   Αφαιρεί τα σχόλια, τα οποία είναι οτιδήποτε ακολουθεί τους χαρακτήρες // σε μία γραμμή, ή οτιδήποτε περιβάλλεται από /* και */ ακόμη και σε πολλές γραμμές.
2) **Δεύτερος Προεπεξεργαστής**
   Ανιχνεύει τον κώδικα του σεναρίου και βρίσκει τις Συναρτήσεις και Καταστάσεις που δηλώνονται σ' αυτό, έτσι ώστε να μπορεί να γίνεται αναφορά σ' αυτές χωρίς να χρειάζεται να δηλωθούν πριν από την κλήση τους. Επίσης, στο βήμα αυτό δηλώνονται αυτόματα οι Μεταβλητές, Συναρτήσεις και Πληροφορίες επέκτασης, όπως θα δούμε πιο κάτω.
3) **Λεξικός / Συντακτικός Αναλυτής**
   Βρίσκει κατά πόσο το σενάριο ακολουθεί τη γραμματική και τους κανόνες της γλώσσας και παράγει το τελικό αποτέλεσμα της ενδιάμεσης γλώσσας.

Το τελικό αποτέλεσμα του μεταγλωττιστή, εάν δεν υπήρξαν λάθη του σεναριογράφου κατά τη μεταγλώττιση, μπορεί αργότερα να διαβαστεί και να εκτελεστεί από τον Εκτελεστή Σεναρίων.

4.2.3 Επέκταση

Για να γράψει κάποιος σενάρια για το Virtual Telemedicine τα οποία να έχουν κάποιο νόημα, χρειάστηκε να γίνει κάποια προσθήκη στις δυνατότητες της γλώσσας. Για παράδειγμα, έπρεπε να υπάρχει κάποιος τρόπος να παρουσιάζεται κάποιο μήνυμα στην οθόνη μέσω του Σεναρίου.

Δε θα ήταν καθόλου σοφό οι δυνατότητες αυτές να ήταν μέρος της γραμματικής της γλώσσας για πολλούς λόγους, με τους κυριότερους να είναι η επεκτασιμότητα και η καλή οργάνωση του έργου. Αν θελήσουμε να προσθέσουμε μία νέα δυνατότητα δε θα πρέπει να αλλάζουμε τον κώδικα του μεταγλωττιστή αλλά του παιχνιδιού, αφού και οι δυνατότητες είναι συγκεκριμένες για την εφαρμογή που χρησιμοποιεί το Scribulance.

Συγκεκριμένα, οι επεκτάσεις είναι τριών ειδών:

- Μεταβλητές οι οποίες δηλώνονται αυτόματα. Ο Επεξεργαστής Σεναρίων έχει τη δυνατότητα να πάρει την τιμή οποιασδήποτε μεταβλητής κατά την εκτέλεση.
- Καθορισμός των πληροφοριών που έχουν τη μορφή κειμένου οι οποίες είναι γνωστές στον Επεξεργαστή Σεναρίων ΠΡΙΝ την εκτέλεση του σεναρίου. Η γλώσσα από μόνη της δεν υποστηρίζει πληροφορίες.
- Συναρτήσεις των οποίων το σώμα ΔΕΝ είναι γραμμένο σε Scribulance αλλά στην ίδια την εφαρμογή που χρησιμοποιεί τον Εκτελεστή Σεναρίων. Όταν στο σενάριο γίνεται κλήση σε τέτοια συνάρτηση, εκτελείται κάποιος κώδικας στην εφαρμογή και επιστρέφεται τελικά μια τιμή στο σενάριο. Εδώ βρίσκεται και η πραγματική δύναμη της Scribulance, διότι μ' αυτό τον τρόπο μπορεί μια συνάρτηση να έχει θεωρητικά απεριόριστες δυνατότητες.

Για περισσότερες πληροφορίες όσον αφορά τις Μεταβλητές, Συναρτήσεις και Πληροφορίες επέκτασης οι οποίες είναι ειδικές για το Virtual Telemedicine, συμβουλευτείτε το **Παράρτημα Α.**

Ο μεταγλωττιστής χρειάζεται να ξέρει ποιες επεκτάσεις είναι ορισμένες. Αυτό δεν το ορίζει ο σεναριογράφος, αλλά ο κατασκευαστής της εφαρμογής που χρησιμοποιεί τη γλώσσα. Αυτό γίνεται αλλάζοντας ένα αρχείο που ονομάζεται **extern.dat** (

Πίνακας 4.1) και βρίσκεται στην ίδια τοποθεσία με τα εκτελέσιμα αρχεία του μεταγλωττιστή. Το αρχείο αυτό έχει την ακόλουθη μορφή:

1. Δηλώσεις Μεταβλητών
   Δηλώνονται όπως ακριβώς και μέσα στο σενάριο
2. Δηλώσεις Συναρτήσεων
   Κάθε δήλωση είναι μ' αυτή τη μορφή: <Όνομα>**$**<Αριθμός Παραμέτρων>**~**
3. Δηλώσεις Πληροφοριών
   Κάθε δήλωση είναι μ' αυτή τη μορφή: $<Όνομα>

Κάθε μια από τις πιο πάνω δηλώσεις είναι προαιρετικές, πρέπει να γίνουν όμως με αυτή τη σειρά.

```
var Message; var Temperature;
var SpO2; var Respiration;
var NIBP1; var NIBP2; var NIBP3;
var IP1a; var IP1b; var IP1c;
var IP2a; var IP2b; var IP2c;
```

```
SetAnimation$1~
SetColor$1~
SetColorVal$3~
SetECG$1~
SetECGto$2~
SetSignal$1~
Random$0~
RandomInt$2~
Round$1~
Floor$1~
Ceiling$1~
Trunc$1~
ToText$1~
ToNumber$1~
WinGame$1~
LoseGame$1~
```

EndGame$2~ GetScore$0~ SetScore$1~ AddScore$1~ Grade$2~
$instructions $history $objectives

Πίνακας 4.1 Το αρχείο extern.dat για την εφαρμογή του Scribulance για το Virtual Telemedicine

4.3 Ο Εκτελεστής Σεναρίων

Ο Εκτελεστής Σεναρίων είναι το πιο βασικό κομμάτι του παιχνιδιού. Λαμβάνει εντολές από τον παίκτη δια μέσω του παιχνιδιού (χωρίς ο παίκτης να το γνωρίζει) και εκτελεί τον κώδικα του σεναρίου όπως αυτό μεταγλωττίστηκε.

4.3.1 Ενδιάμεση γλώσσα

Η Ενδιάμεση Γλώσσα είναι μια παραλλαγμένη μορφή του αρχικού κώδικα. Είναι πολύ δύσκολο να γραφτεί απευθείας από άνθρωπο, όμως μπορεί να διαβάζεται εύκολα από κάποιον υπολογιστή. Αυτό γιατί κάθε εντολή που γράφει ένας σεναριογράφος αντιστοιχεί σε πολλές εντολές. Για παράδειγμα, η εντολή:

```
Number = 5 * 6 + 3;
```

είναι στην καλύτερη περίπτωση 6 ατομικές εντολές:

1) Το 5 μπαίνει στη μνήμη
2) Το 6 μπαίνει στη μνήμη
3) Γίνεται πολλαπλασιασμός
4) Το 3 μπαίνει στη μνήμη
5) Γίνεται πρόσθεση
6) Γίνεται ανάθεση στη μεταβλητή Number

Εκτός από εντολές, στην ενδιάμεση γλώσσα περιγράφονται και πληροφορίες που αφορούν τη ροή εκτέλεσης του σεναρίου, όπως για παράδειγμα οι προτάσεις **if.**

Επομένως, ο εκτελεστής μόλις λάβει ένα αρχείο το περνά από ένα Αναγνώστη. Ο Αναγνώστης υπάγεται στον Εκτελεστή και περνά από δύο διακριτά στάδια:

1) **Η Διαμοίραση του σεναρίου**
2) **Μετάφραση ατομικών εντολών**

4.3.2 Διαμοίραση του Σεναρίου

Το σενάριο διαμοιράζεται αρχικά σε υποπρογράμματα. Ένα υποπρόγραμμα είναι μία ακέραια ακολουθία εντολών που εκτελείται κυριολεκτικά. Είναι, δηλαδή, οι δηλώσεις μεταβλητών και οι ορισμοί των συναρτήσεων και γεγονότων. Ο εκτελεστής τρέχει ένα υποπρόγραμμα μέχρι να τερματίσει, εκτός αν ο σεναριογράφος το τερματίσει πρόωρα με την εντολή `return;`

Η θέση των υποπρογραμμάτων σημειώνεται από τον Αναγνώστη χρησιμοποιώντας πίνακες κατακερματισμού και άλλες βοηθητικές δομές ανάλογα με τον τύπο του υποπρογράμματος. Για παράδειγμα, όταν αργότερα θα γίνει κλήση σε μια συνάρτηση που όρισε ο σεναριογράφος, ο Εκτελεστής θα ξέρει που ακριβώς βρίσκεται αυτή η συνάρτηση με βάση τον πίνακα κατακερματισμού.

Μια ροή περιγράφεται ως μια συγκεκριμένη εκτέλεση των εντολών. Ένα υποπρόγραμμα δεν τρέχει απαραίτητα με τον ίδιο τρόπο. Αυτό γιατί η ροή διακλαδώνεται με τη χρήση των προτάσεων `if […] then […] else […]` (το `else […]` είναι προαιρετικό). Όταν υπάρχει τέτοιος έλεγχος συνθήκης, ο Αναγνώστης σημειώνει που πρέπει να συνεχίσει η ροή όταν ισχύει ή δεν ισχύει η συνθήκη ανάλογα.

Για να μπορέσει ο Αναγνώστης να διαμερίσει το σενάριο, κάθε πρόγραμμα γραμμένο στην Ενδιάμεση Γλώσσα περιέχει αναγκαστικά επισημάνσεις (

Πίνακας 4.2). Κάθε επισήμανση ξεκινά με τον χαρακτήρα @ για να ξεχωρίζει με τις πραγματικές εντολές. Όχι όλες οι επισημάνσεις είναι χρήσιμες για τον Αναγνώστη, μερικές υπάρχουν για σκοπούς αποσφαλμάτωσης.

Επισήμανση	**Περιγραφή**
`INFO <Name>`	Ακολουθεί κείμενο το οποίο ορίζει τις πληροφορίες ενός σεναρίου. Το Name για το Virtual Telemedicine μπορεί να είναι ένα από τα Objectives, Instructions ή History, όμως μια άλλη εφαρμογή μπορούσε να δέχεται διαφορετικές πληροφορίες.
`FNOTE <Name> <Args>`	Επισημαίνει ότι στο σενάριο υπάρχει ο ορισμός συνάρτησης με όνομα Name και αριθμό παραμέτρων Args
`FEXTERN <Name> <Args>`	Επισημαίνει πως το σενάριο παρέχει υποστήριξη για την συνάρτηση επέκτασης με όνομα Name και αριθμό παραμέτρων Args.
`FUNC <Name>`	Επισημαίνει την αρχή του κώδικα της συνάρτησης με όνομα Name.
`STATE <Name>`	Επισημαίνει την αρχή μιας Κατάστασης.
`ENDSTATE`	Επισημαίνει το τέλος μιας Κατάστασης.
`SCENARIO $<Desc>`	Περιγράφει το Description του σεναρίου.
`STARTWITH <Name>`	Η αρχική κατάσταση.
`ENTER / ENDENTER`	Ορίζουν την αρχή και τέλος αντίστοιχα του κώδικα για το γεγονός εισόδου μιας κατάστασης.
`EXIT / ENDEXIT`	Ορίζουν την αρχή και τέλος αντίστοιχα του κώδικα για το γεγονός εξόδου μιας κατάστασης.
`OPTION $<Text>`	Ορίζει μια Επιλογή με κείμενο Text.
`OPIF / CHECKOPIF`	Ορίζει την αρχή και το τέλος αντίστοιχα της προϋπόθεσης Επιλογής, αν υπάρχει.
`NOOPIF`	Επισημαίνει πως η Επιλογή δεν έχει προϋπόθεση.
`ENDOPTION`	Επισημαίνει το τέλος του κώδικα μιας επιλογής.
`COND`	Επισημαίνει την αρχή ενός Γεγονότος Συνθήκης.
`CHECKCOND`	Επισημαίνει το τέλος της Συνθήκης.
`ENDCOND`	Επισημαίνει το τέλος του Γεγονότος Συνθήκης.

TIME <mins> <secs>	Επισημαίνει την αρχή ενός Γεγονότος Χρόνου.
ENDTIME	Επισημαίνει το τέλος του Γεγονότος Χρόνου.

Πίνακας 4.2: Επισημάνσεις της ενδιάμεσης γλώσσας

4.3.3 Μετάφραση ατομικών εντολών

Για να μπορούν να εκτελεστούν οι εντολές, μπαίνουν με τη σωστή σειρά σε μια λίστα από αντικείμενα τύπου `StoredCommand`. Κάθε StoredCommand περιέχει μια λίστα από παραμέτρους και δείκτη σε μια συνάρτηση υλοποιημένη σε C#, η οποία εκτελεί την εντολή με βάσει τις παραμέτρους.

Για να μπορούν να εκτελούνται οι μαθηματικές πράξεις με τη σωστή προτεραιότητα, ο μεταγλωττιστής έχει ήδη μετατρέψει την κάθε πράξη σε μια ακολουθία ατομικών εντολών (Πίνακας 4.3) που ακολουθούν Postfix Notation, λόγω της σωστής χρήσης των κανόνων γραμματικής που ακολουθεί.

Έτσι, ο Εκτελεστής ορίζει δύο στοίβες. Στη μία στοίβα μπαίνουν μέσα οι τιμές και στην άλλη οι διάφοροι τελεστές. Όταν πρέπει να εφαρμοστεί μία δυαδική πράξη, ο τελεστής που βρίσκεται στην κεφαλή της Στοίβας Τελεστών αφαιρείται, όπως επίσης και οι δύο τελεστές που βρίσκονται στην κεφαλή της Στοίβας Τιμών. Ο Εκτελεστής ξέρει ποια πράξη να εκτελέσει ανάλογα με τον τελεστή. Το αποτέλεσμα της πράξης αυτής μπαίνει τελικά στη Στοίβα Τιμών.

Οι μοναδιαίοι τελεστές λειτουργούν με διαφορετικό τρόπο. Δεν μπαίνουν στη Στοίβα Τελεστών και όταν πρέπει να εφαρμοστούν, εφαρμόζονται στην κεφαλή της Στοίβας Τιμών.

Στο τέλος κάθε πράξης, είμαστε βέβαιοι ότι λαμβάνουμε το σωστό αποτέλεσμα, το οποίο τοποθετείται στη Στοίβα Τιμών ως το μοναδικό αποτέλεσμα.

Η Στοίβα Τιμών χρησιμοποιείται και για κλήση συναρτήσεων με παραμέτρους. Όταν μια συνάρτηση καλείται, οι παράμετροι που περνούν μπαίνουν στη Στοίβα Τιμών. Α-

κολούθως, η συνάρτηση ορίζει μεταβλητές με τα ονόματα των παραμέτρων με τέτοια σειρά ώστε οι κάθε παράμετρος να λάβει τη σωστή τιμή.

Αν μια συνάρτηση επιστρέφει κάποια τιμή, τότε τοποθετείται στη στοίβα τιμών η τιμή αυτή που επιστρέφει. Αν όχι, τότε τοποθετείται αυτόματα η σταθερά τιμή `null`.

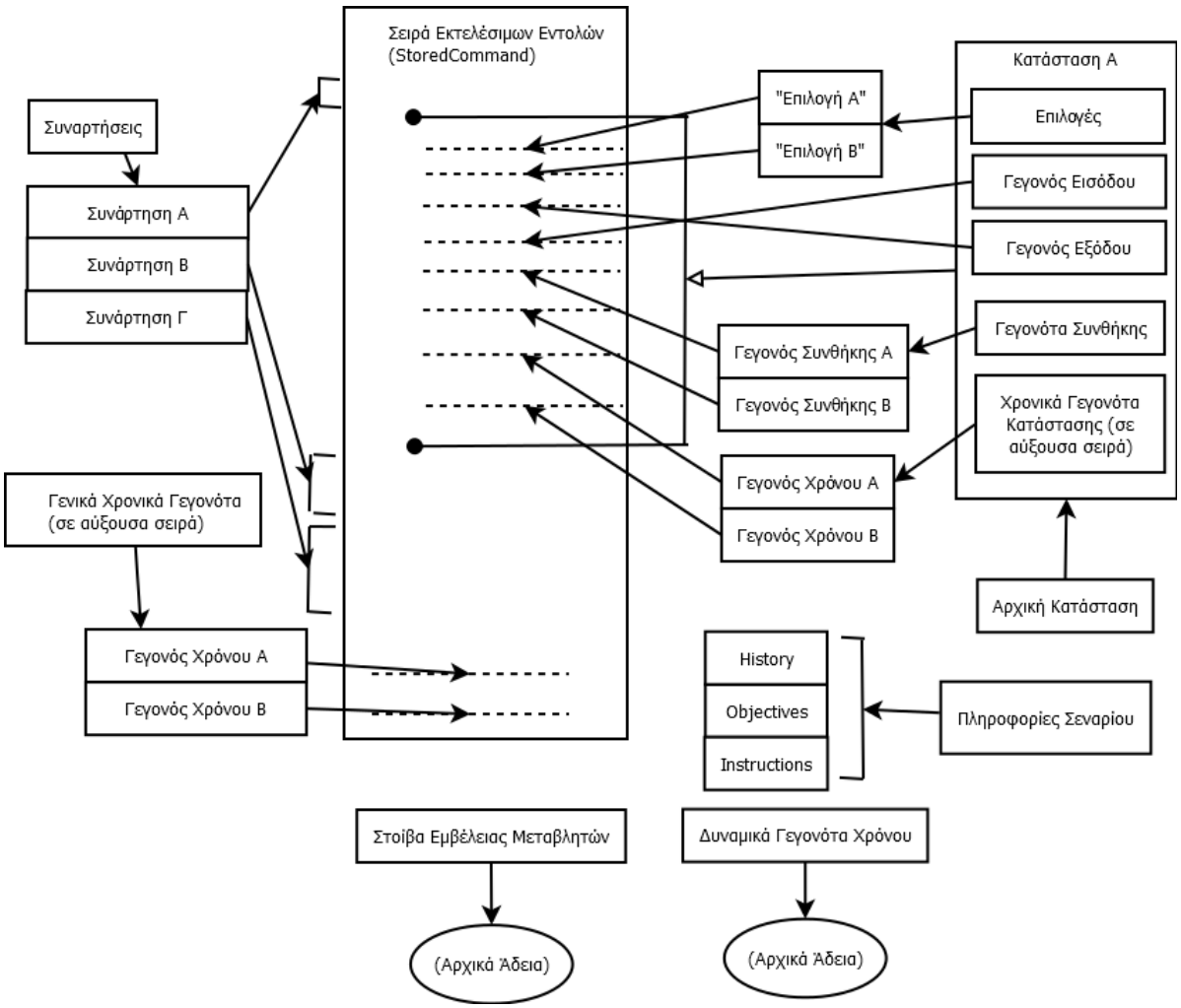
Εντολή	**Παράμετροι**	**Περιγραφή**
`Declare <Name>`	`Name`	Δήλωση μεταβλητής με όνομα <Name>.
`EnterScope`		Εισαγωγή σε νέα Εμβέλεια (`{`).
`ExitScope`		Έξοδος Εμβέλειας (`}` ή `return`).
`Assign <Name>`	`Name`	Ανάθεση από την κεφαλή της Στοίβα Τιμών στη μεταβλητή με όνομα <Name>
`Apply <Op>`	`Op`	Εφαρμογή μοναδιαίου τελεστή με όνομα Op στην κεφαλή της Στοίβας Τιμών.
`PopOp`	–	Εκτέλεση της πράξης του τελεστή που βρίσκεται στην κεφαλή της Στοίβας Τελεστών.
`PopVal`	–	Αφαίρεση της κεφαλής της Στοίβας Τιμών.
`PushVar <Name>`	`Name`	Αποτίμηση της μεταβλητής με όνομα Name και τοποθέτηση της τιμής της στην κεφαλή της Στοίβας Τιμών.
`PushOp <Op>`	`Op`	Τοποθέτηση του τελεστή με όνομα Op στη Στοίβα Τελεστών.
`PushLit <Type> $<Value>`	`Value`	Τοποθέτηση στην κεφαλή της Στοίβας Τιμών σταθεράς με τιμή Value.
`Return`	–	Επιστροφή στο προηγούμενο υποπρόγραμμα. Αν δεν υπάρχει τέτοιο, τότε ο Εκτελεστής Σεναρίων τερματίζει την παρούσα εκτέλεση και περιμένει περαιτέρω οδηγίες από το παιχνίδι.
`GOTO <State>`	`State`	Μετακινεί τη ροή της εκτέλεσης στην Κατάσταση με όνομα State, αφού τρέξει

		τα γεγονότα εξόδου και εισόδου της παρούσας / επόμενης Κατάστασης. Οι στοίβες Τιμών και Τελεστών και όλες οι πληροφορίες ροής μηδενίζονται.
JUMPTO <State>	State	Όπως πιο πάνω, με τη διαφορά ότι δεν εκτελούνται τα γεγονότα εξόδου και εισόδου.
CALL <Function>	Function	Κλήση συνάρτησης ορισμένης από τον Σεναριογράφο. Μετακινεί τη ροή της εκτέλεσης στη θέση της συνάρτησης με όνομα Function.
EXTERNAL <Function>	Function	Κλήση συνάρτησης επέκτασης ορισμένης από το παιχνίδι. Εκτελεί κώδικα ορισμένο από τον κατασκευαστή της εφαρμογής που χρησιμοποιεί τον Εκτελεστή Σεναρίων.
DYNBIND <Mins> <Secs>	StoredCommand Minutes Seconds	Εκτέλεση εντολής StoredCommand η οποία είναι η προηγούμενη εντολή, σε χρόνο Minutes + Seconds . Η προηγούμενη εντολή σημειώνεται ως κενή.
CHECKIF	False Index	Ελέγχει την κεφαλή της Στοίβας Τιμών για το αποτέλεσμα της συνθήκης που ελέγχεται. Αν η συνθήκη είναι αληθής, τότε η ροή του κώδικα μετακινείται κανονικά στην επόμενη εντολή. Αν όχι, τότε μετακινείται στην κατάλληλη θέση όπως ορίζεται από το False Index,
IFFALSE	Skip Index	Βρίσκεται αμέσως πριν από το False Index ενός CHECKIF και μετακινεί τη ροή στο Skip Index έτσι ώστε να μην εκτελεστεί ο κώδικας ψευδής συνθήκης που ακολουθεί ένα else.

Πίνακας 4.3: Οι ατομικές εντολές της Ενδιάμεσης Γλώσσας

4.3.4 Ροή εκτέλεσης ενός Σεναρίου

Κατά την ανάγνωση ενός σεναρίου, δημιουργείται ένα σύμπλεγμα δομών (Σχήμα 4.11) που περιέχει όλες τις πληροφορίες που απαιτούνται για να τρέξει ένα σενάριο. Υπενθυμίζεται πως οι συναρτήσεις, οι καταστάσεις, οι πληροφορίες και τα γεγονότα είναι προαιρετικά. Ένα σενάριο μεταγλωττίζει ακόμη και χωρίς συναρτήσεις ή γεγονότα. Απαιτείται να υπάρχει τουλάχιστο μία Κατάσταση, έστω και άδεια.



Σχήμα 4.11: Δομή που απαιτείται από τον Εκτελεστή των σεναρίων

Τα γεγονότα χρόνου είναι ταξινομημένα σε αύξουσα σειρά διότι κάθε δευτερόλεπτο ελέγχεται αν ένα απ' αυτά πρέπει να εκτελεστεί. Η ταξινόμηση τους καθιστά αυτόν τον έλεγχο πιο εύκολο, διότι απλώς ελέγχεται το επόμενο γεγονός.

Τα δυναμικά γεγονότα είναι επίσης μια ταξινομημένη λίστα με γεγονότα τα οποία παραπέμπουν σε εντολές οι οποίες ΔΕΝ υπάρχουν μέσα στη Σειρά Εκτελέσιμων Εντολών. Οι εντολές αυτές κάνουν κλήση σε συνάρτηση (είτε ορισμένη είτε επέκτασης). Δημιουργούνται δυναμικά κατά την εκτέλεση του σεναρίου, και καταστρέφονται μετά την ολοκλήρωση τους.

Κατά την αρχή ενός σεναρίου ξεκινά ο κώδικας από την αρχή, όπου είναι δηλωμένες όλες οι γενικές μεταβλητές του σεναρίου. Μόλις τελειώσει η δήλωση, εκτελείται ο κώδικας που βρίσκεται στην αρχή της αρχικής κατάστασης, όπου γίνεται η δήλωση των μεταβλητών της κατάστασης. Ακολούθως, εκτελείται το Γεγονός Εισόδου της αρχικής κατάστασης.

Εφόσον δεν εκτελείται κώδικας του σεναρίου, η ροή εκτέλεσης επιστρέφει στο παιχνίδι, το οποίο είναι εκτός της δικαιοδοσίας του Εκτελεστή. Το σενάριο συνεχίζει την εκτέλεση του όταν χρειάζεται να εκτελεστεί κάποιο γεγονός, όταν δηλαδή φθάσει η ώρα να εκτελεστεί κάποιο γεγονός χρόνου, ή έχει ικανοποιηθεί μια συνθήκη, ή ο χρήστης έχει πατήσει σε κάποια επιλογή.

Κατά την εκτέλεση συναρτήσεων επέκτασης, η ροή μπαίνει και πάλι στο παιχνίδι, όπου είναι ορισμένη η συνάρτηση, ακολούθως όμως επιστρέφει αμέσως στον Εκτελεστή κανονικά.

Κατά τη εκτέλεση εντολών, υπάρχει ένας δείκτης ο οποίος κινείται σειριακά προς τα κάτω. Ο δείκτης όμως μπορεί να μετακινηθεί όταν:

- Μια συνθήκη σε πρόταση **if** είναι ψευδής, οπόταν πρέπει να αγνοήσει το μπλοκ που εκτελείται όταν είναι αληθής.
- Μια συνθήκη σε πρόταση **if** είναι αληθής και υπάρχει μπλοκ **else,** οπόταν πρέπει να το αγνοήσει.
- Γίνει κλήση σε συνάρτηση ορισμένη από τον σεναριογράφο, οπόταν:

1. Βρίσκεται η συνάρτηση από το λεξικό συναρτήσεων.
2. Μετακινείται ο δείκτης στο σημείο στο οποίο ξεκινά η εκτέλεση της.

- Τερματίσει μια συνάρτηση, οπόταν ο δείκτης μετακινείται στην εντολή που βρισκόταν αμέσως μετά από την κλήση της συνάρτησης
- Γίνει αλλαγή της κατάστασης με την εντολή **goto.** Σ' αυτή την περίπτωση, ο δείκτης μετακινείται στο Γεγονός Εξόδου (αν υπάρχει) της τρέχουσας κατάστασης. Αφού αυτό γίνει τότε ο δείκτης μετακινείται στην αρχή της νέας κατάστασης, όπου δηλώνονται οι μεταβλητές της. Μετά ο δείκτης μετακινείται στο Γεγονός Εισόδου της νέας κατάστασης, αν αυτό υπάρχει, και τέλος η ροή επιστρέφει στο παιχνίδι. Οποιαδήποτε ροή του Εκτελεστή υπήρχε ήδη διαγράφεται. (Κάθε κατάσταση έχει όλες τις πληροφορίες για τη θέση των μελών που την αποτελούν).
- Γίνει αλλαγή της κατάστασης με την εντολή **jumpto.** Τα βήματα που ακολουθούνται είναι όπως πιο πάνω, με τη διαφορά ότι αγνοούνται τα γεγονότα Εισόδου και Εξόδου για την τρέχουσα και επόμενη κατάσταση αντίστοιχα.

4.3.5 Μεταβλητές

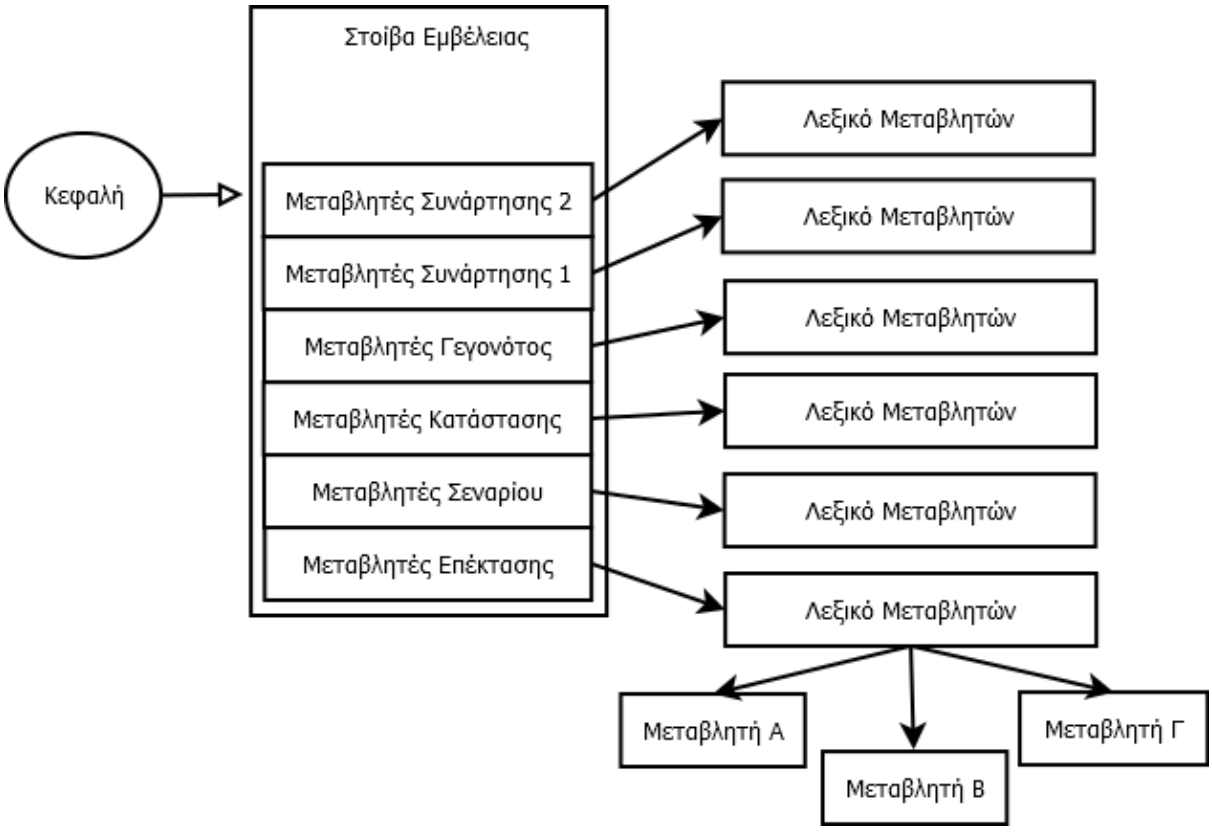
Η δήλωση μεταβλητών γίνεται μόνο:

- Πριν το σενάριο (Μεταβλητές Επέκτασης)
- Στην αρχή του Σεναρίου, μετά τις Πληροφορίες
- Στην αρχή μιας Κατάστασης, πριν τις Επιλογές και Γεγονότα
- Στην αρχή μιας Συνάρτησης, Επιλογής ή Γεγονότος

Επομένως υπάρχουν πολλές ομάδες μεταβλητών. Μια ομάδα ονομάζεται και εμβέλεια μεταβλητών, διότι περιέχει μεταβλητές οι οποίες ισχύουν μόνο για περιορισμένες γραμμές κώδικα. Για παράδειγμα, κατά την αλλαγή μιας κατάστασης, όλες οι μεταβλητές που ανήκαν σ' αυτή διαγράφονται γιατί η ροή βρίσκεται εκτός της εμβέλειας της κατάστασης. Όλες οι ομάδες περιέχονται σε μια στοίβα που λέγεται Στοίβα Μεταβλητών (Σχήμα 4.12).

Ακόμη και αν υπάρξει αναδρομική κλήση συνάρτησης, δημιουργείται νέο λεξικό για την τρέχουσα κλήση. Επίσης, οι μεταβλητές σεναρίου και επέκτασης δεν βγαίνουν ποτέ εκτός εμβέλειας παρά μόνο εάν τερματιστεί το σενάριο.

Όταν γίνεται αποτίμηση μιας μεταβλητής, ελέγχεται η εμβέλεια που βρίσκεται μέσα στην εμβέλεια που βρίσκεται στην κεφαλή της στοίβας. Αν αυτή δεν βρεθεί, τότε μπορεί να υπάρχει στις μεταβλητές Κατάστασης, αν και μόνο αν η αναφορά στη μεταβλητή βρίσκεται απευθείας σε Γεγονός ή Επιλογή της κατάστασης. Αν όχι, τότε υπάρχει είτε στις μεταβλητές σεναρίου, είτε στις μεταβλητές επέκτασης. Λόγω του ότι ο μεταγλωττιστής απαγορεύει χρήση μεταβλητών χωρίς να έχουν οριστεί, αν γίνεται αναφορά σε μεταβλητή τότε σίγουρα έχει δηλωθεί.



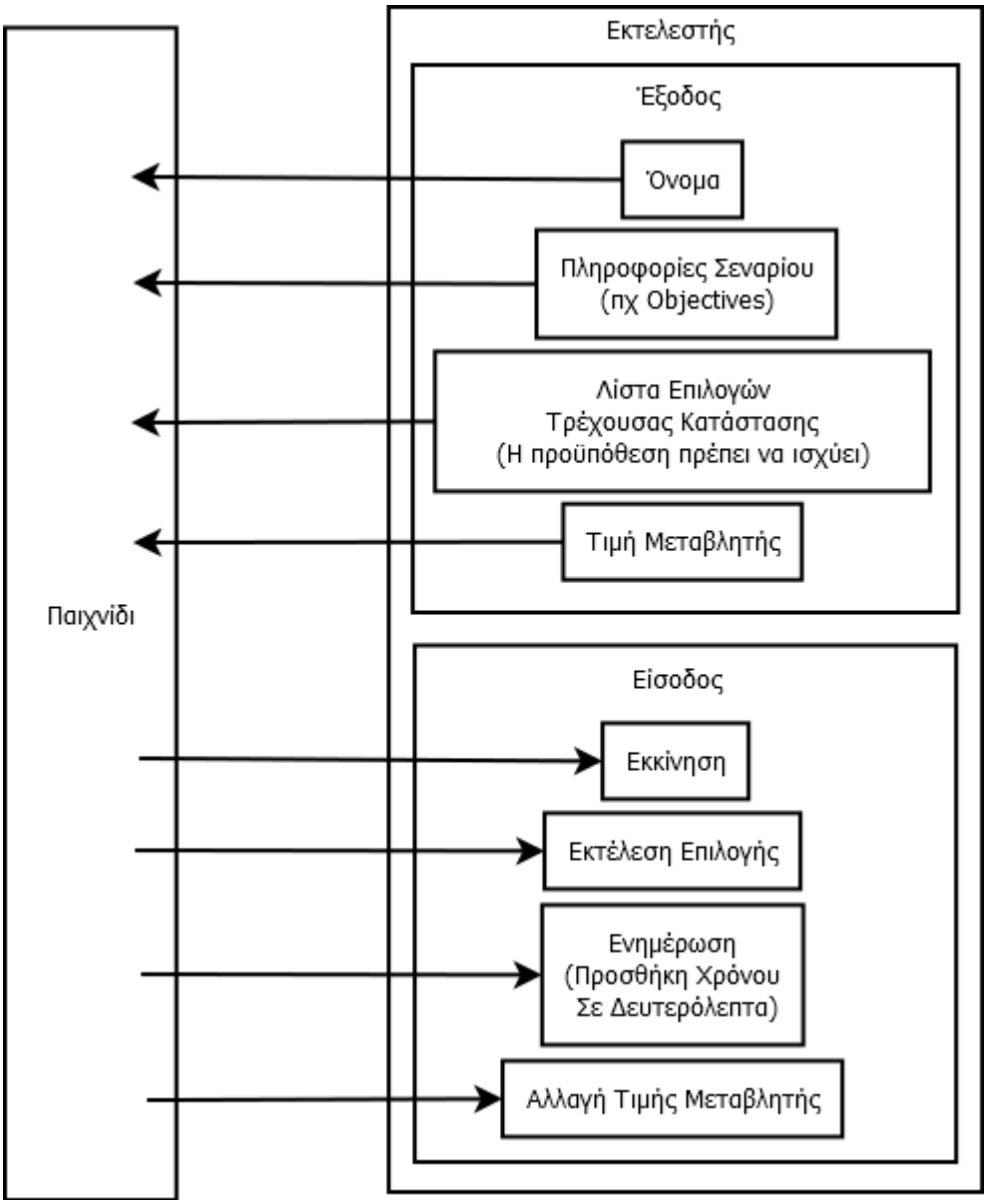
Σχήμα 4.12: Η Στοίβα Μεταβλητών

4.3.6 Διαπροσωπεία μεταξύ του Εκτελεστή και του Παιχνιδιού

Η εσωτερική λειτουργία του Εκτελεστή είναι κρυμμένη από το Παιχνίδι, ή οποιαδήποτε άλλη εφαρμογή που θα μπορούσε να τον χρησιμοποιεί. Έτσι υπάρχει μια διαπροσωπεία

(Σχήμα 4.13) που ενώνει τα δύο μέρη, αποτρέποντας τη λανθασμένη χρήση του Εκτελεστή.

Για να μπορέσει ο Εκτελεστής να τρέξει τις συναρτήσεις επέκτασης, πρέπει πρώτα να εγγραφούν στο Αρχείο Καταγραφής Συναρτήσεων. Αυτό γίνεται κατά την εκκίνηση του Παιχνιδιού και όχι κατά την εκτέλεση ή ανάγνωση ενός σεναρίου.
Για να καταγραφεί μια συνάρτηση πρέπει να χρησιμοποιηθεί η κλάση `FunctionRegistry`, η οποία παρέχει τη στατική μέθοδο Register, η οποία λαμβάνει σαν παραμέτρους το όνομα, τον αριθμό παραμέτρων και δείκτη σε κάποια μέθοδο που είναι ήδη υλοποιημένη. Η μέθοδος για να γίνεται δεκτή πρέπει να μπορεί να λάβει απεριόριστο αριθμό παραμέτρων (ασχέτως πως η ίδια η συνάρτηση παίρνει πεπερασμένο αριθμό παραμέτρων) τύπου Value, και να επιστρέφει μια τιμή τύπου Value.



Σχήμα 4.13: Διαπροσωπεία του Εκτελεστή με το Παιχνίδι

Value είναι μία τιμή όπως ορίζεται από τη γλώσσα, η οποία ουσιαστικά είναι μία κλάση η οποία ενθυλακώνει μια δυναμική τιμή που μπορεί να είναι:

- Κενή (`null` ή `none`)

  Η τιμή δεν έχει οριστεί. Μπορεί για παράδειγμα να είναι η αποτίμηση μιας μεταβλητής που δεν αρχικοποιήθηκε, ή το αποτέλεσμα μιας συνάρτησης που δεν επιστρέφει τιμή.
- Απροσδιόριστη (`illegal`)

Αποτέλεσμα πράξης η οποία δεν ορίζεται, όπως για παράδειγμα σύγκριση κειμένου με αριθμό.

- Αριθμός
  Είτε ακέραιος, είτε πραγματικός.
- Κείμενο
- Λογική Τιμή
  `True` ή `False`

4.4 Πληροφόρηση προς τον παίκτη

Λόγω του εκπαιδευτικού περιεχόμενου του παιχνιδιού, είναι σημαντικό ο χρήστης να παίρνει τις πληροφορίες που χρειάζεται από το παιχνίδι. Πρέπει επίσης όμως ο κάθε σεναριογράφος να έχει τη δυνατότητα να παρουσιάσει τις πληροφορίες αυτές εύκολα, χρησιμοποιώντας μόνο τη Γλώσσα Σεναρίων, χωρίς να χρειάζεται να επέμβει στην υλοποίηση του παιχνιδιού.

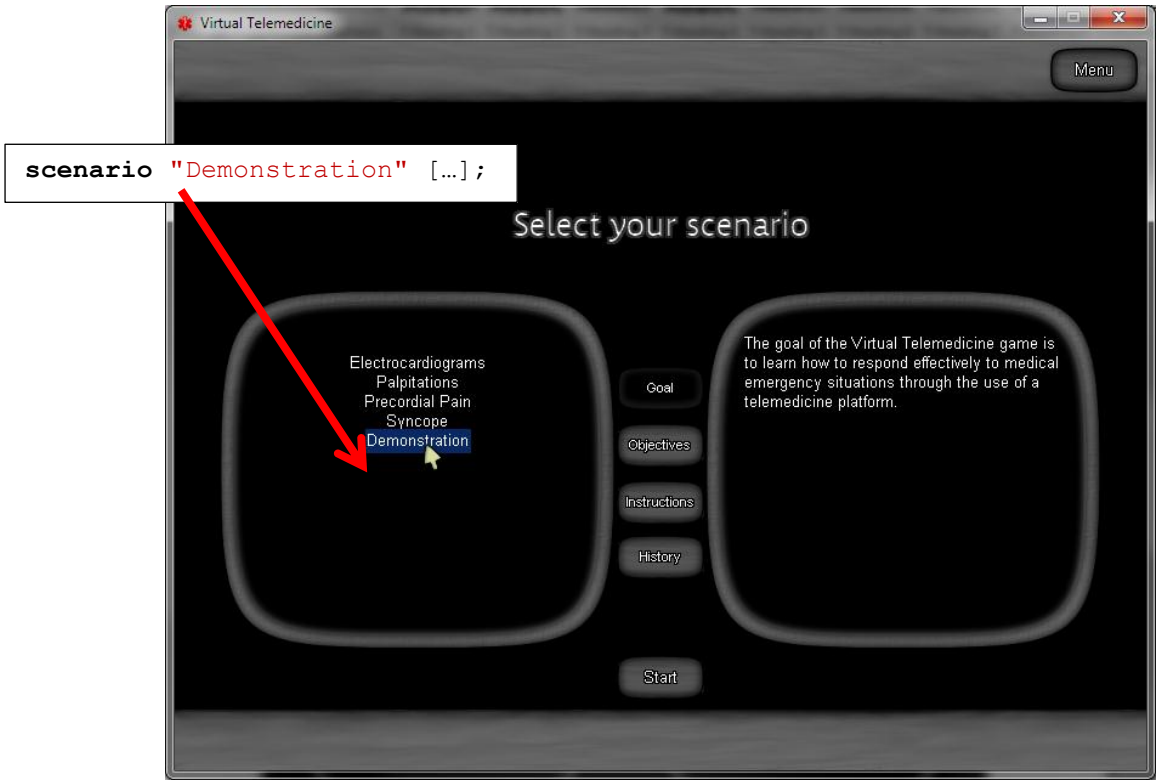
Αυτός ο σκοπός ήταν και το κίνητρο πίσω από το σύστημα επέκτασης, δηλαδή τις πληροφορίες σεναρίου και μεταβλητές / συναρτήσεις επέκτασης.

Η πρώτη πληροφορία που βλέπει ο παίκτης είναι η Περιγραφή (description) του Σεναρίου. Ακολουθεί η δήλωση Πληροφοριών επέκτασης. Μια Πληροφορία μπορεί να είναι μέχρι και τα τρία από τα: History, Objectives και Instructions. Το καθένα απ' αυτά εμφανίζεται στην οθόνη επιλογής σεναρίου (Σχήμα 4.15, Σχήμα 4.16 και Σχήμα 4.17,)

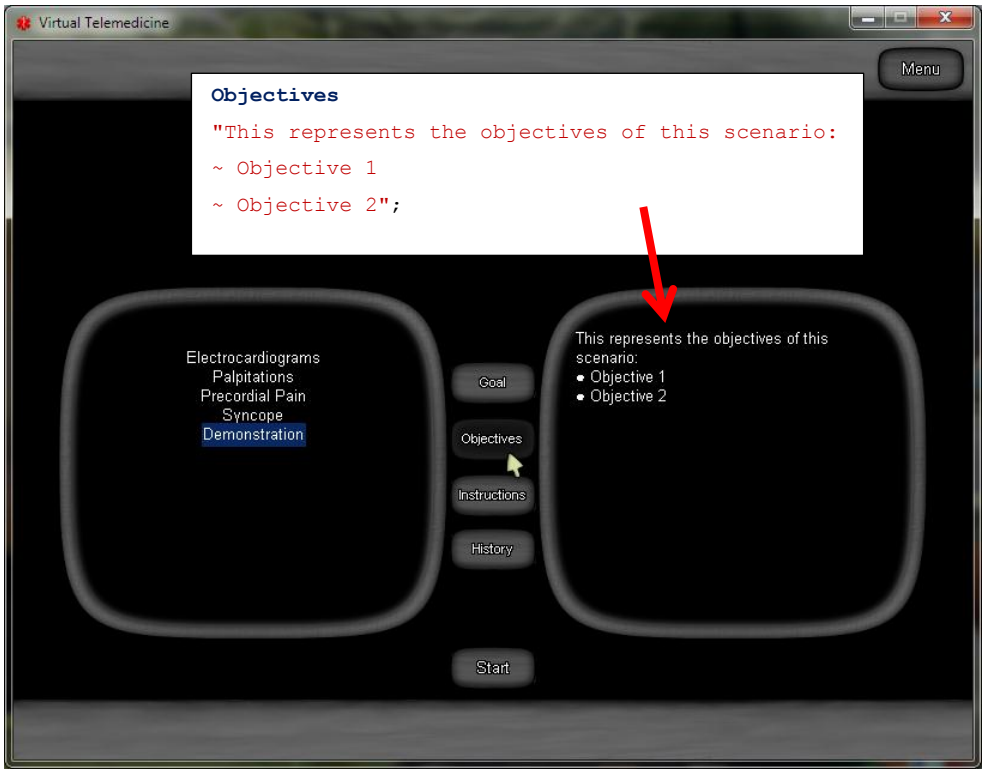
```
scenario "Demonstration" first_state;
History "This represents the patient's history.
This text can span across multiple lines";
Objectives
"This represents the objectives of this scenario:
~ Objective 1
~ Objective 2";
```

```
Instructions
"This represents the instructions for this scenario:
~ Step 1
~ Step 2
~ Step 3";

// Ακολουθεί η δήλωση των μεταβλητών σεναρίου
```

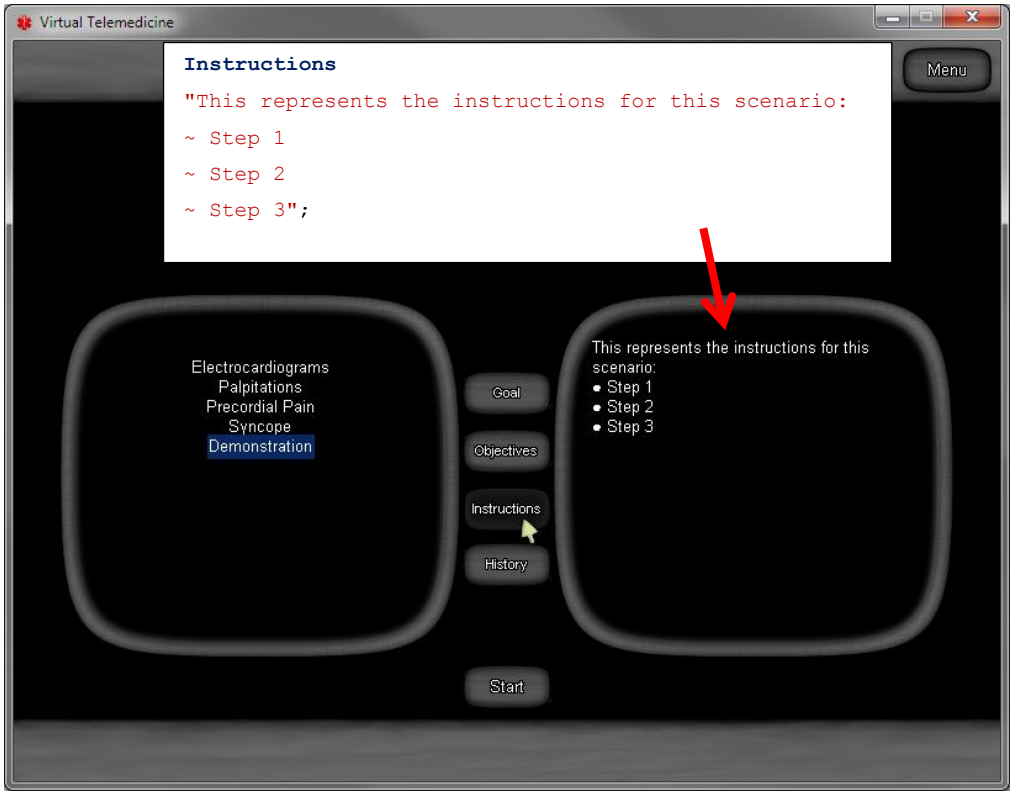
Πίνακας 4.4: Δήλωση Πληροφοριών του Σεναρίου στη γλώσσα σεναρίων Scribulance



Σχήμα 4.14: Επιλογή του σεναρίου Demonstration



Σχήμα 4.15: Επιλογή των Objectives

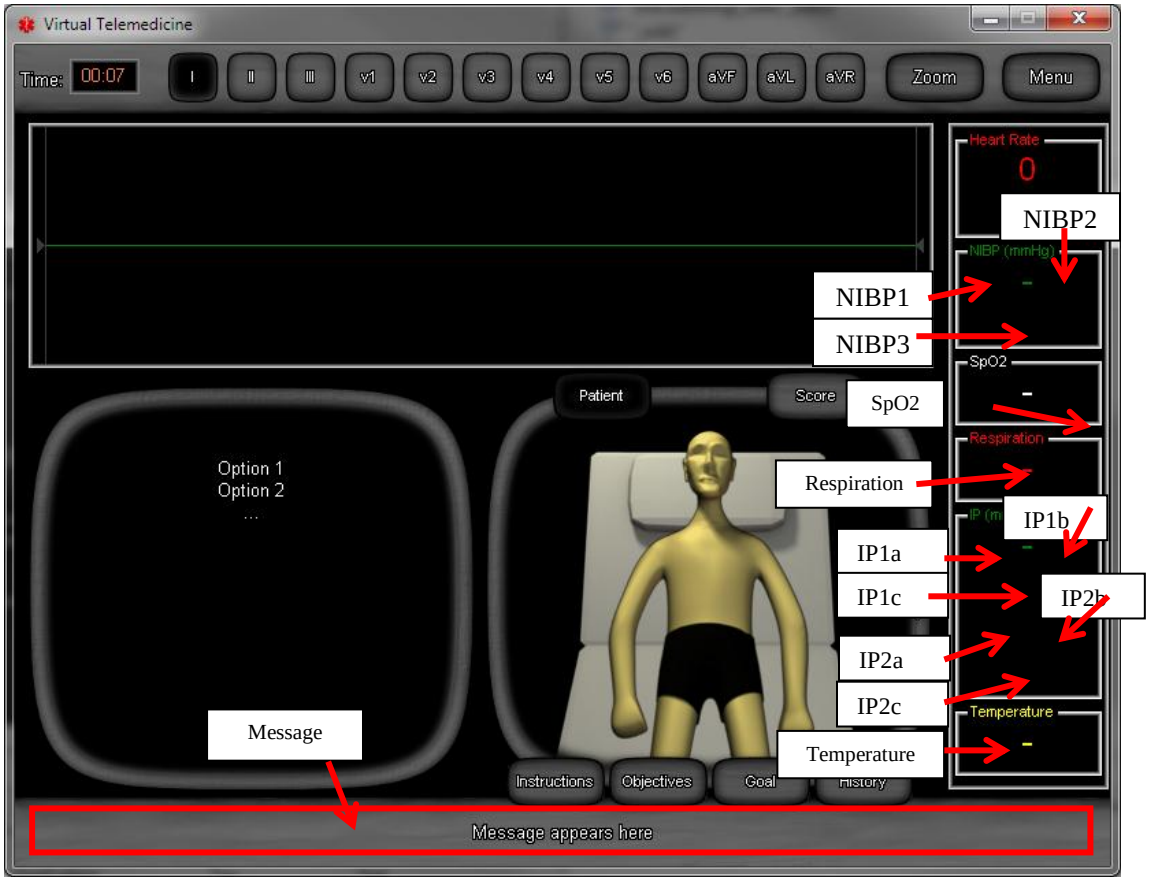


Σχήμα 4.16: Επιλογή των Instructions

Σχήμα 4.17: Επιλογή του History

Αφού τρέξει το σενάριο, οι μεταβλητές επέκτασης που υποστηρίζει το παιχνίδι εμφανίζονται στην οθόνη. Ο σεναριογράφος έχει στη διάθεση του τις ακόλουθες μεταβλητές, οι οποίες εμφανίζονται άμεσα στην οθόνη (Σχήμα 4.18):

- Message
- NIBP1
- NIBP2
- NIBP3
- Sp02
- Respiration
- IP1a
- IP1b
- IP1c
- IP2a
- IP2b
- IP2c
- Temperature



Σχήμα 4.18: Μεταβλητές Επέκτασης και ο χώρος εμφάνισης τους

Χρησιμοποιώντας συναρτήσεις επέκτασης, ο σεναριογράφος μπορεί να παρουσιάσει περισσότερες πληροφορίες.

- **SetAnimation**(«όνομα»)

  Θέτει την κίνηση του ασθενή με το όνομα που δίδεται. Το όνομα μπορεί να είναι ένα από τα ακόλουθα:

  - "BreatheHard"
  - "BreatheNormal"
  - "BreatheSlow"
  - "Dead"
  - "Default"
  - "Headache"
  - "Stomachache"

- **SetColor**(«όνομα»)

Θέτει το χρώμα του ασθενή με το όνομα που δίδεται. Το όνομα μπορεί να είναι ένα από τα ακόλουθα:

- "Normal"
- "Pale"
- "Blue"

- **SetColorVal**(«κόκκινο», «πράσινο», «μπλε»)

  Θέτει το χρώμα του ασθενή σε μορφή Red Green Blue, όπου η τιμή για κάθε ένα από τα τρία χρώματα είναι μια τιμή από το 0 μέχρι το 1. Για παράδειγμα το μαύρο είναι (0, 0, 0), το άσπρο (1, 1, 1), το κόκκινο (1, 0, 0), το πράσινο (0, 1, 0) και το μπλε (0, 0, 1). Οποιαδήποτε άλλη τιμή συνδυάζει τα χρώματα αυτά για να παράγει αυτό που επιθυμεί ο χρήστης.

- **SetECG**(«όνομα»)

  Θέτει το καρδιογράφημα (Σχήμα 4.19) σε ένα από τα ακόλουθα:

  - "120"
  - "150"
  - "180"
  - "30"
  - "45"
  - "60"
  - "75"
  - "90"
  - "A.FIB"
  - "A.FLUTTER"
  - "A.PAUSE"
  - "ARRHYTHMIA"
  - "ARTEFACT"
  - "AUTO"
  - "AV-BL.I"
  - "AV-BL.II"
  - "BIGEM"
  - "COUPLET"
  - "DEM.PACER"
  - "INTERF"

- "NONE"
- "PACER"
- "PACER.MAL"
- "R-ON-T"
- "RBBB"
- "RUN"
- "SIN.ARRHY"
- "ST.DEPR"
- "ST.ELEV"
- "SVPB.S"
- "V-FIB"
- "V-RHYTHM"
- "VBP.MULT"
- "VBP.UNI"
- "VTACH"

- **SetECGto**(«όνομα», «προσαρμογή»)

  Θέτει το καρδιογράφημα (Σχήμα 4.19) σε ένα από τα πιο πάνω, αλλά εφαρμόζει προσαρμογή σε κάποια συγκεκριμένη τιμή. Για παράδειγμα, το καρδιογράφημα St.Elev είναι σε 75 κτύπους και μ' αυτή τη συνάρτηση θα εμφανιστεί στο ρυθμό που θα δίνεται από την προσαρμογή.

- **WinGame**(«μήνυμα»)

  Τελειώνει το παιχνίδι με νίκη για τον παίκτη, εμφανίζοντας (μαζί με άλλες πληροφορίες) και το «μήνυμα»

- **LoseGame**(«μήνυμα»)

  Όπως πιο πάνω, με τη διαφορά ότι παίκτης χάνει το παιχνίδι

- **EndGame**(«όριο», «μήνυμα»)

  Όπως πιο πάνω, με τη διαφορά ότι ο παίκτης κερδίζει το παιχνίδι μόνο εάν η βαθμολογία του είναι μεγαλύτερη ή ίση από το όριο.

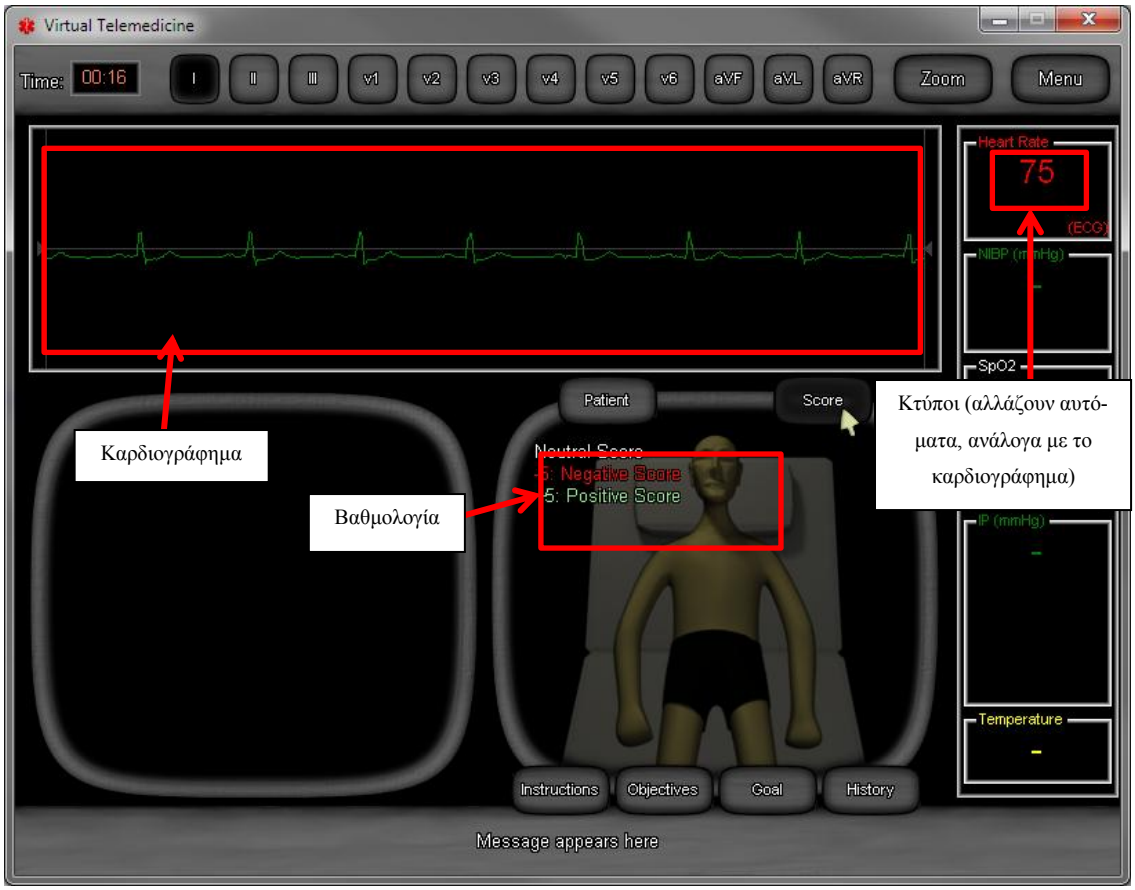
- **Grade**(«βαθμός», «μήνυμα»)

  Προσθέτει, ή αφαιρεί αν ο βαθμός είναι αρνητικός, κάποιο βαθμό από τη βαθμολογία (Σχήμα 4.19) του παίκτη. Το «μήνυμα» εμφανίζεται εάν ο παίκτης πατήσει στο κουμπί Score, αλλά εμφανίζεται και στο τέλος του παιχνιδιού. Το μή-

νυμα εμφανίζεται κόκκινο, αν ο βαθμός είναι αρνητικός, πράσινο, αν είναι θετικός, ή άσπρο αν ο βαθμός είναι μηδέν.

- **SetScore**(«βαθμός»)
  Θέτει τη βαθμολογία χωρίς να το εμφανίσει στο χρήστη. Χρήσιμο για την αρχική βαθμολογία του σεναρίου.

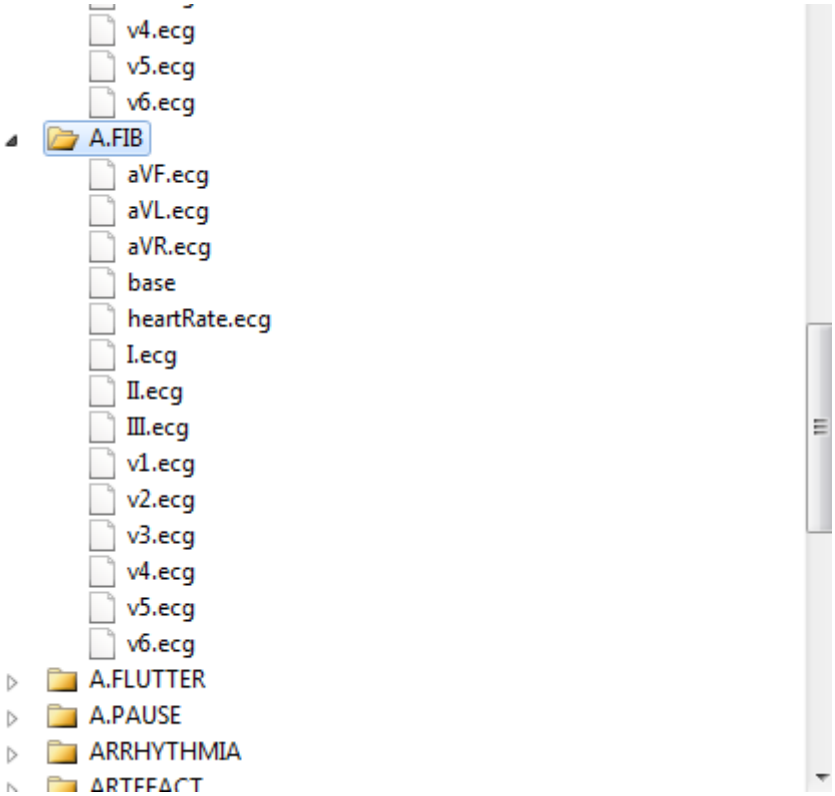
Υπάρχουν και άλλες συναρτήσεις επέκτασης, όμως δεν εμφανίζουν κάποια άμεση αλλαγή προς τον χρήστη. Για περισσότερες πληροφορίες, αναφερθείτε στο Παράρτημα Α'.



Σχήμα 4.19: Χώρος εμφάνισης του καρδιογραφήματος και της βαθμολογίας

4.5 Καρδιογραφήματα

Τα καρδιογραφήματα που χρησιμοποιεί το παιχνίδι είναι φυλαγμένα σε αρχεία .ecg. Τα αρχεία αυτά κωδικοποιούν αριθμούς κινητής υποδιαστολής σε ψηφιακή μορφή που βρίσκονται σε σειρά. Κάθε γράφημα υπάρχει σε μορφή φάκελου που περιέχει .ecg αρχεία (Σχήμα 4.20). Σε κάθε φάκελο υπάρχει ένα αρχείο για τους κτύπους και ένα για κάθε ένα από τα 12 σήματα.

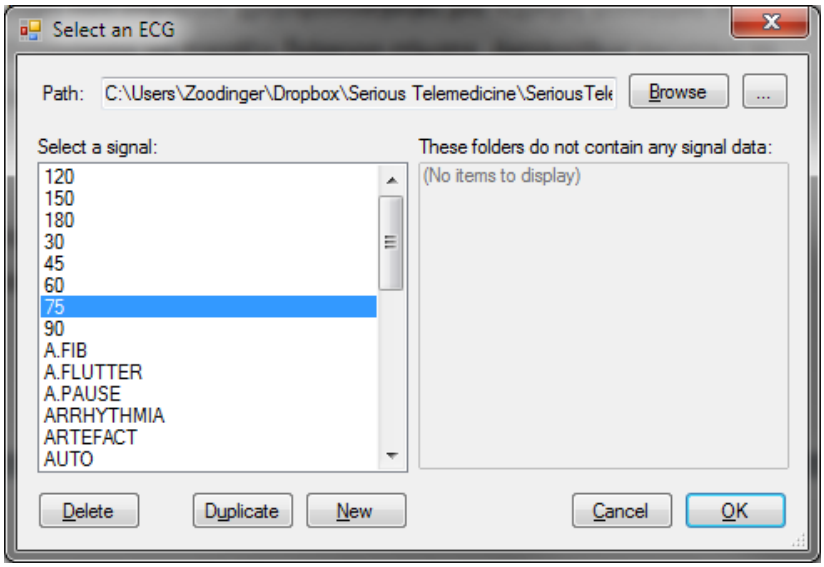


Σχήμα 4.20: Οργάνωση των γραφημάτων ECG

Κάθε αριθμός βρίσκεται σε απόσταση 10 χιλιοστών δευτερολέπτου από τον επόμενο. Στην περίπτωση των κτύπων, ο αριθμός αυτός είναι ο αριθμός που εμφανίζεται στον παίκτη ως ο ρυθμός κτύπων της καρδιάς. Στην περίπτωση των σημάτων, οι τιμές είναι σε μVolts.

Για την καταγραφή των σημάτων χρησιμοποιήθηκε μία τεχνητή γεννήτρια παλμών, η οποία είχε τη δυνατότητα να παράγει διάφορα σήματα. Ακολούθως ενώσαμε τη γεννήτρια με ένα μηχάνημα καταγραφής σημάτων και πήραμε τις τιμές του κάθε σήματος.

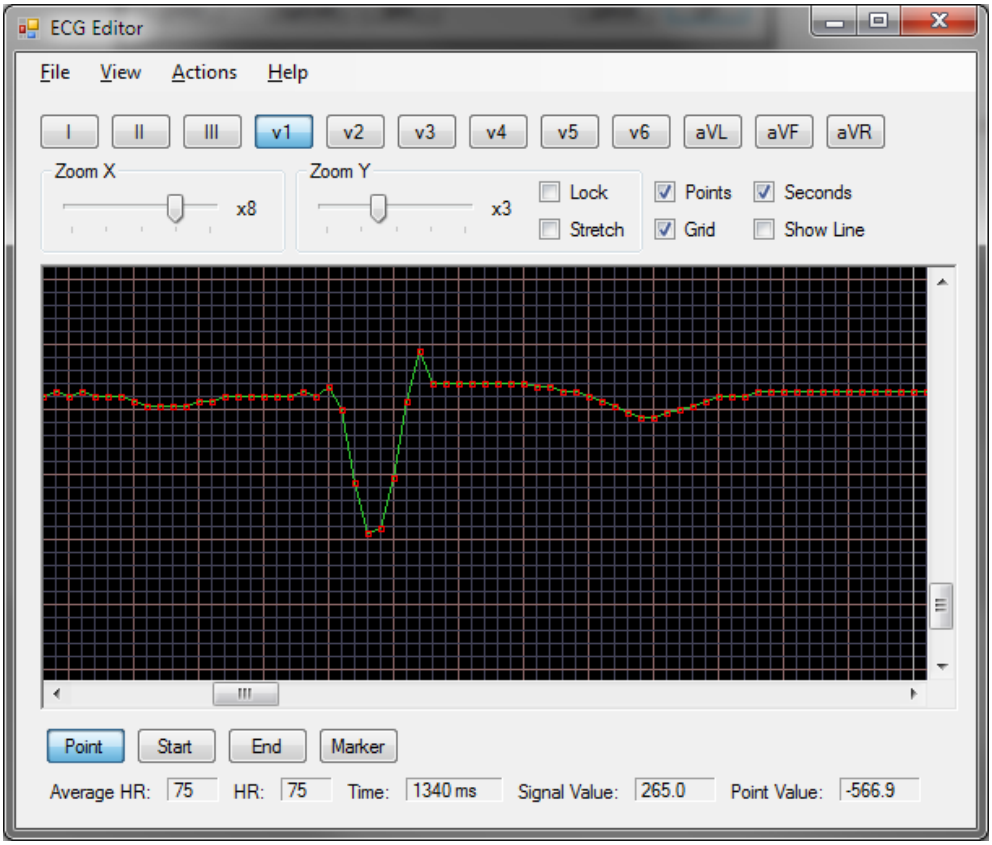
Κάποια σήματα όμως, όπως το RBBB (Right Bundle Branch Block) δεν υποστηρίζονταν από τη γεννήτρια. Επίσης, στο παιχνίδι τα σήματα πρέπει να επαναλαμβάνονται. Λόγω του ότι όλα τα σήματα καταγράφτηκαν στα 10 δευτερόλεπτα, τα σήματα δεν επαναλαμβάνονταν κυκλικά σωστά. Γι' αυτό το σκοπό γράφτηκε ένα πρόγραμμα το οποίο επέτρεπε την αλλαγή και επεξεργασία των σημάτων (Σχήμα 4.22, Σχήμα 4.23). Μόλις ξεκινήσει το πρόγραμμα, εμφανίζεται ένα παράθυρο το οποίο σου επιτρέπει να αντιγράψεις ή τροποποιήσεις ένα υπάρχον καρδιογράφημα, ή να δημιουργήσεις ένα νέο (Σχήμα 4.21).



Σχήμα 4.21: Επιλογέας γραφήματος

Το πρόγραμμα δεν είναι μέρος του «Virtual Telemedicine», αλλά χρησιμοποιήθηκε για να εξαλείψει τα δύο προβλήματα που αναφέρθηκαν πιο πάνω. Γράφτηκε σε C# με το framework της .NET και μπορεί να επαναχρησιμοποιηθεί και για άλλες εργασίες.

Σχήμα 4.22: Το πρόγραμμα επεξεργασίας γραφημάτων



Σχήμα 4.23: Μεγέθυνση του γραφήματος ώστε να είναι πιο εύκολη η επεξεργασία του

Κεφάλαιο 5

Το «Virtual Telemedicine» ως Σοβαρό Παιχνίδι

5.1 Εισαγωγή στους μηχανισμούς του παιχνιδιού

Το παιχνίδι ανταποκρίνεται στην ανάγκη να εκπαιδεύσει τους γιατρούς για την επίλυση προβλημάτων σε πραγματικές καταστάσεις ιατρικής περίθαλψης μέσω ενός συστήματος τηλεϊατρικής. Ο χρήστης σε ένα πρωτότυπο σενάριο, ένας εν ενεργεία γιατρός, καλείται να ανταποκριθεί στις ιατρικές ανάγκες ενός εικονικού ασθενή που βρίσκεται εξ αποστάσεως με τη χρήση ενός συστήματος τηλεϊατρικής. Ο χρήστης λαμβάνει κάποιες ενέργειες αφού μελετήσει προσεκτικά τις πληροφορίες που του δίνονται από το σύστημα, οι οποίες είναι βασισμένες στις πραγματικές πληροφορίες τις οποίες θα μπορούσε να λάβει σε ένα πραγματικό σενάριο. [3]

5.2 Η έννοια του Σεναρίου

Ένα Σενάριο ορίζει μια κατάσταση στην οποία θα μπορούσε ένας χρήστης να βρεθεί σαν ιατρός. Περιγράφει την ιστορία του ασθενή και την κατάσταση του, ποιες ενέργειες πρέπει να λάβει ο χρήστης, καθώς και κάποιες λανθασμένες ενέργειες που θα μπορούσε να πάρει. Κάθε ενέργεια οδηγεί σε κάποιο αποτέλεσμα, το οποίο μπορεί να είναι είτε θετικό είτε αρνητικό.

Το παιχνίδι στην beta μορφή περιέχει τρία σενάρια [2]:

- ***Procardial Pain,*** στο οποίο ο ασθενής αντιμετωπίζει ένα οξύ στεφανιαίο σύνδρομο.
- ***Palpitations,*** στο οποίο ο ασθενής βιώνει καρδιακές αρρυθμίες.
- ***Syncope,*** στο οποίο ο ασθενής έχει διαγνωστεί με μια κολποκοιλιακό αποκλεισμό.

Ο χρήστης μπορεί εύκολα να προσθέσει νέα σενάρια, τα οποία μπορεί να γράψει είτε ο ίδιος είτε ένας τρίτος.

5.3 Πώς το Virtual Telemedicine είναι σοβαρό παιχνίδι

Για να δείξουμε πως το Virtual Telemedicine είναι σοβαρό παιχνίδι, πρέπει πρώτα να δείξουμε ότι είναι παιχνίδι και όχι απλή προσομοίωση. Αυτό θα το δείξουμε χρησιμοποιώντας τον ορισμό του παιχνιδιού που αναφέραμε προηγουμένως [5], αναγνωρίζοντας τα τρία στοιχεία που πρέπει να το αποτελούν:

1. **Ένα τεχνητό εμπόδιο**

   Στην οθόνη υπάρχει ένας εικονικός ασθενής, ο οποίος έχει την άμεση ανάγκη ιατρικής βοήθειας. Ο ασθενής δεν είναι πραγματικός και επομένως οποιοδήποτε παίξιμο δεν δύναται να βλάψει κανένα πρόσωπο. Το εμπόδιο είναι δηλαδή η κατάσταση του ασθενή, την οποία ο παίκτης-γιατρός πρέπει να αντιμετωπίσει

2. **Ένα σύνολο από κανόνες**

   Στη γενική περίπτωση, οι κανόνες είναι απλοί: Ο χρήστης πρέπει απλώς να επιλέξει τη σωστή ενέργεια που αρμόζει στην κατάσταση. Οι υπόλοιποι κανόνες καθορίζονται από το σενάριο. Για παράδειγμα, ένα σενάριο μπορεί να έχει όριο χρόνου, ή όριο λάθος επιλογών.

3. **Ένα μετρήσιμο αποτέλεσμα**

   Ένα σενάριο μπορεί να τελειώσει είτε σε επιτυχία, είτε σε αποτυχία. Πέρα από αυτό το αποτέλεσμα όμως, ο σεναριογράφος μπορεί να ορίσει ειδική βαθμολογία που εξαρτάται από τις επιλογές του χρήστη. Η βαθμολογία αυτή παρουσιά-

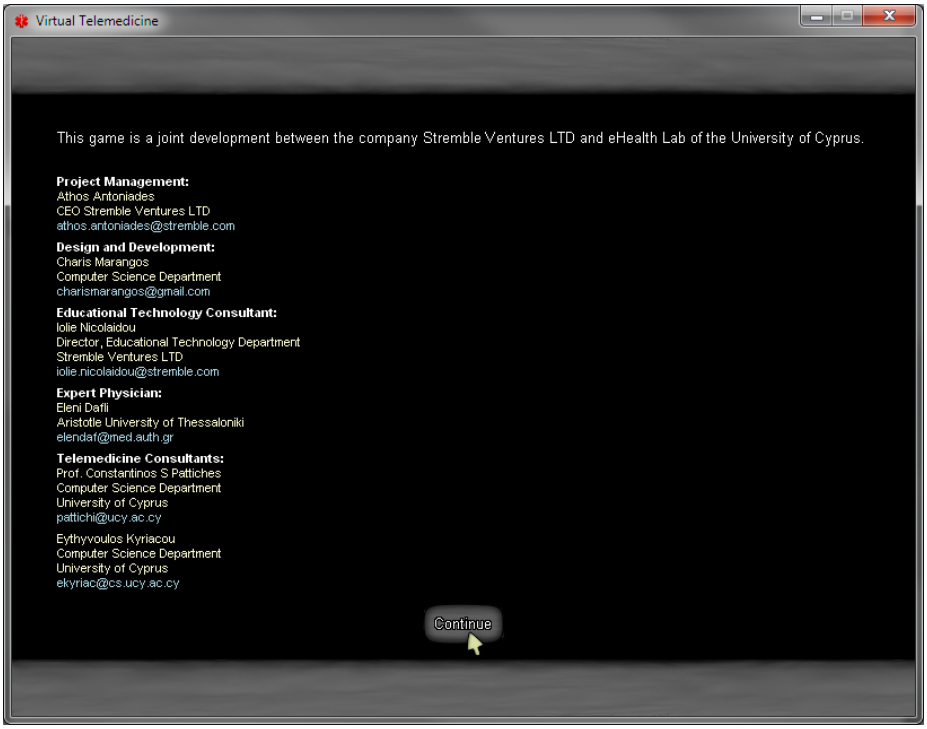
ζεται στον τερματισμό του σεναρίου, όμως είναι και προσβάσιμη κατά τη διάρκεια του παιχνιδιού.

Το παιχνίδι είναι και Σοβαρό Παιχνίδι, διότι ο παίκτης υιοθετεί τον ρόλο ενός γιατρού. Οι διάφορες καταστάσεις στις οποίες μπορεί να βρίσκεται ο ασθενής είναι πραγματικές. Κατ' επέκταση, οι ορθές αποφάσεις που μπορεί να λάβει είναι ιατρικά ορθές και δε θυσιάζεται η ρεαλιστικότητα για χάρη της ψυχαγωγίας. Παίζοντας το παιχνίδι ένας θα μπορούσε να μάθει πως να συμπεριφερθεί σε καταστάσεις παρόμοιες με αυτές του σεναρίου, ή να ανανεώσει τις γνώσεις του.

5.4 Οδηγίες χρήσης

Ακολουθώντας κάποιες οδηγίες, ο χρήστης θα βρει την εμπειρία του παιξίματος πιο εύκολη.

Μόλις τρέξει το παιχνίδι, αντιμετωπίζει την οθόνη πληροφοριών (Σχήμα 5.1). Πατώντας το κουμπί *Continue*, ο χρήστης θα βρει μπροστά του το Κυρίως Μενού (Main Menu)



Σχήμα 5.1: Οθόνη πληροφοριών

5.4.1 Κυρίως Μενού

Το Κυρίως Μενού έχει δύο μορφές. Η πρώτη μορφή (Σχήμα 5.2) εμφανίζεται μόλις ο χρήστης πάει για πρώτη φορά στο Κυρίως Μενού, ή αμέσως μόλις επιστρέψει στο Κυρίως Μενού αφού τελειώσει ένα παιχνίδι. Η δεύτερη μορφή (Σχήμα 5.3) εμφανίζεται όταν ο χρήστης επιλέξει να πάει στο Κυρίως Μενού κατά τη διάρκεια ενός παιχνιδιού.
Οι επιλογές που αντιμετωπίζει ο χρήστης είναι:

- **Resume**

  Ο χρήστης μπορεί να επιλέξει το *Resume* όταν πήγε στο Κυρίως Μενού κατά τη διάρκεια ενός παιχνιδιού. Σ' αυτή την περίπτωση το παιχνίδι παγώνει, και με την επιλογή του *Resume* ο χρήστης συνεχίζει κανονικά το παιχνίδι.
- **New Game**

  Ο χρήστης μπορεί από εδώ να ξεκινήσει νέο παιχνίδι, αρχίζοντας ένα σενάριο από την αρχή.
- **Restart**

  Ο χρήστης μπορεί να επιλέξει το *Restart* όταν πήγε στο Κυρίως Μενού κατά τη διάρκεια ενός παιχνιδιού. Πατώντας το Restart ο χρήστης ξεκινά το τρέχον σενάριο από την αρχή.
- **Scenario Manager**

  Αυτή η επιλογή παίρνει τον χρήστη στον Διαχειριστή Σεναρίων, απ' όπου μπορεί να προσθέσει ή αφαιρέσει νέα σενάρια.
- **About**

  Αυτή η επιλογή παίρνει τον χρήστη στην αρχική οθόνη πληροφοριών που εμφανίζεται μόλις ξεκινήσει το παιχνίδι.
- **Exit**

  Αυτή η επιλογή τερματίζει το παιχνίδι.

Σχήμα 5.2: Το Κυρίως Μενού στην αρχική του μορφή

Σχήμα 5.3: Το Κυρίως Μενού στη δεύτερη μορφή

5.4.2 Επιλογή Σεναρίων

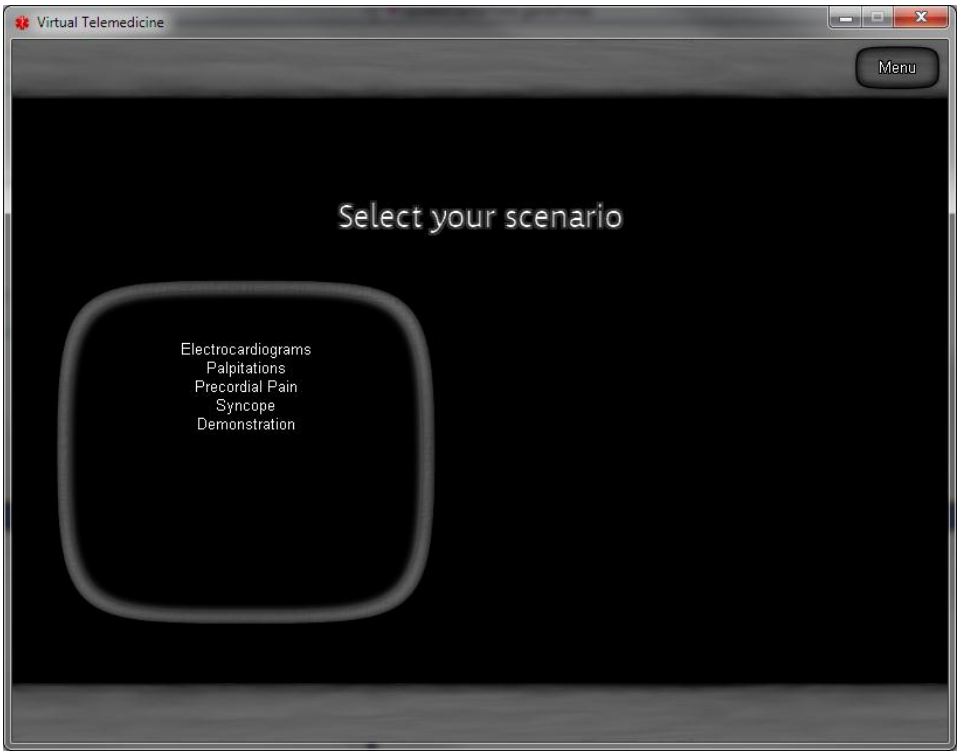
Όταν ο χρήστης πατήσει στην επιλογή *New Game* στο Κυρίως Μενού, παρουσιάζεται σ' αυτόν η οθόνη επιλογής σεναρίων (Σχήμα 5.4), η οποία περιέχει τη λίστα σεναρίων.

Η λίστα αυτή περιέχει όλα τα σενάρια τα οποία είναι άμεσα διαθέσιμα. Αυτά τα σενάρια είναι:

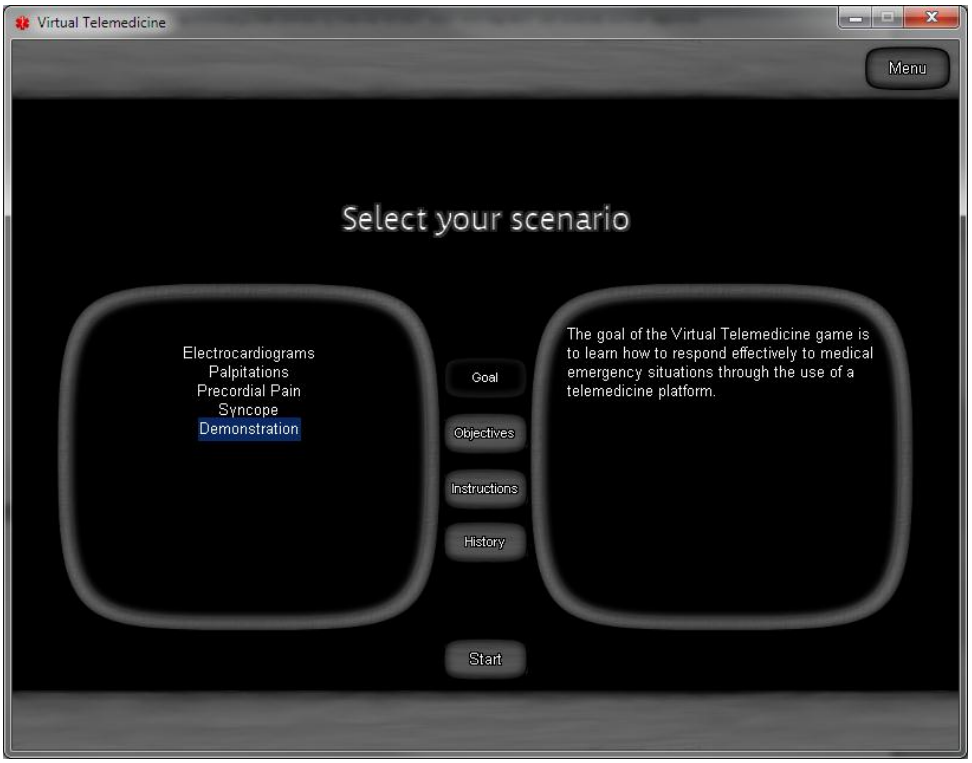
- Τα τρία σενάρια που υπάρχουν με την εγκατάσταση του παιχνιδιού: ***Palpitations***, ***Precordial Pain*** και ***Syncope***
- Το σενάριο ***Electrocardiograms***, το οποίο επίσης υπάρχει με την εγκατάσταση του παιχνιδιού και βρίσκεται για σκοπούς επίδειξης διάφορων ηλεκτροκαρδιογραφημάτων που μπορεί ένας χρήστης να συναντήσει κατά τη διάρκεια του παιχνιδιού
- Οποιαδήποτε άλλα σενάρια προστεθούν από την οθόνη Διαχείρισης Σεναρίων. Σ' αυτή την περίπτωση το επιπλέον σενάριο είναι μόνο ένα, το ***Demonstration***, το οποίο χρησιμοποιείται αποκλειστικά για σκοπούς επίδειξης για αυτή την εργασία.

Πατώντας πάνω σε ένα από αυτά τα σενάρια, εμφανίζονται στο χρήστη οι πληροφορίες για το επιλεγόμενο σενάριο (Σχήμα 5.5). Όταν ένα σενάριο είναι επιλεγμένο, εμφανίζονται και επιπλέον επιλογές: *Goal*, *Objectives*, *Instructions* και *History*. Η επιλογή *Goal* είναι η ίδια για όλα τα σενάρια. Οι επιλογές *Objectives*, *Instructions* και *History* μπορούν να διαφέρουν και ορίζονται από τον σεναριογράφο.

Αφού ένα σενάριο είναι επιλεγμένο, πατώντας το κουμπί *Start* ο χρήστης θα ξεκινήσει να παίζει το σενάριο αυτό. Αν ο χρήστης θελήσει να μετακινηθεί πίσω στο Κυρίως Μενού, μπορεί να πατήσει το κουμπί *Main Menu*.



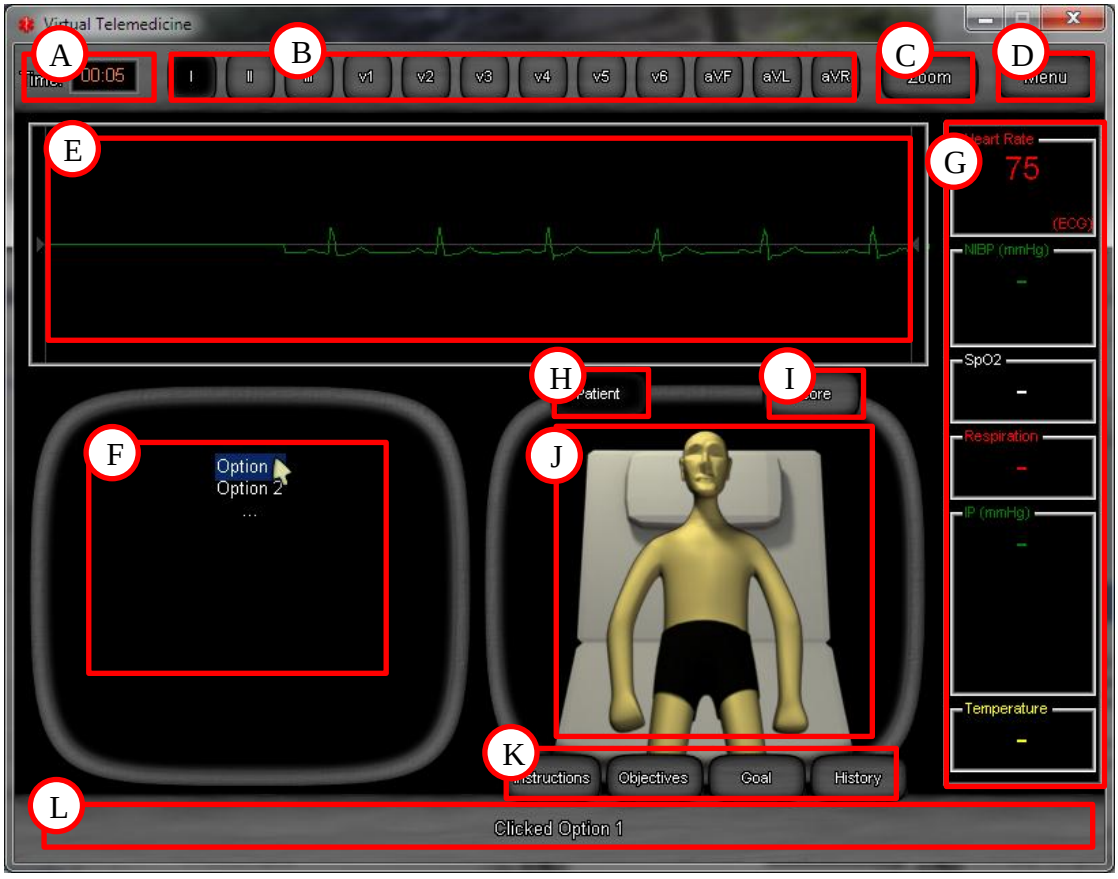
Σχήμα 5.4: Οθόνη επιλογής σεναρίου



Σχήμα 5.5: Ένα σενάριο είναι επιλεγμένο και εμφανίζονται επιπλέον επιλογές

5.4.3 Επεξήγηση Διαπροσωπείας παιχνιδιού

Με το ξεκίνημα ενός σεναρίου, παρουσιάζεται στο χρήστη μια οθόνη η οποία έχει τη μορφή όπως φαίνεται στο Σχήμα 5.6.



Σχήμα 5.6: Το γραφικό περιβάλλον του Virtual Telemedicine

Ακολουθεί επεξήγηση για την κάθε περιοχή:

A. Ο χρόνος διεξαγωγής του παιχνιδιού, σε δευτερόλεπτα.
B. Επιλογή σήματος για το ηλεκτροκαρδιογράφημα.
C. Όταν αυτό το κουμπί είναι πατημένο, το ηλεκτροκαρδιογράφημα μεγεθύνεται αυτόματα ώστε να χωρέσει την περιοχή που εμφανίζεται (Σχήμα 5.8). Ο χρήστης θα χρειαστεί να ενεργοποιήσει αυτή την εντολή όταν θελήσει να εξετάσει το ηλεκτροκαρδιογράφημα πιο προσεκτικά. Διαφορετικά, το σήμα εμφανίζεται σε κλίμακα από -6000μV μέχρι +6000μV (Σχήμα 5.7).
D. Πατώντας αυτό το κουμπί ο χρήστης θα μετακινηθεί πίσω στο Κυρίως Μενού.
E. Σ' αυτή την περιοχή εμφανίζεται το ηλεκτροκαρδιογράφημα του ασθενή.

F. Εδώ υπάρχουν όλες οι επιλογές που έχει στη διάθεση του ο γιατρός για μια δεδομένη στιγμή. Όταν μια επιλογή εμφανίζεται με γκρίζα γράμματα, τότε η εντολή αυτή είναι προσωρινά μη εφικτή.

G. Εδώ παρουσιάζονται όλες οι πληροφορίες του ασθενή, όπως και στην οθόνη επιλογής σεναρίου. Αυτές δίνονται μέσα από το σενάριο.

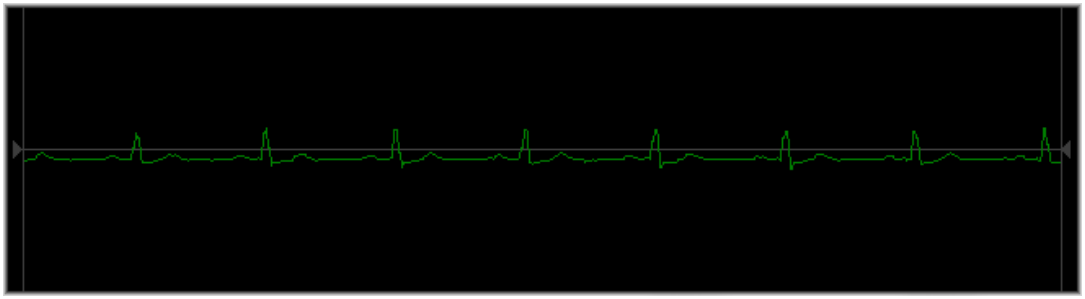
H. Όταν αυτό το κουμπί είναι επιλεγμένο, οι εικόνα του ασθενή φαίνεται καθαρά όπως πιο πάνω.

I. Όταν αυτό το κουμπί είναι επιλεγμένο, εμφανίζεται πάνω από τον ασθενή η βαθμολογία (Σχήμα 5.9). Ο ασθενής σ' αυτή την περίπτωση εμφανίζεται πιο σκούρος, έτσι ώστε το κείμενο να είναι πιο εμφανές. Οι θετικές βαθμολογίες εμφανίζονται πράσινες, ενώ οι αρνητικές εμφανίζονται κόκκινες. Όταν μια βαθμολογία είναι ουδέτερη, τότε λειτουργεί σαν σημείωση και εμφανίζεται άσπρη.

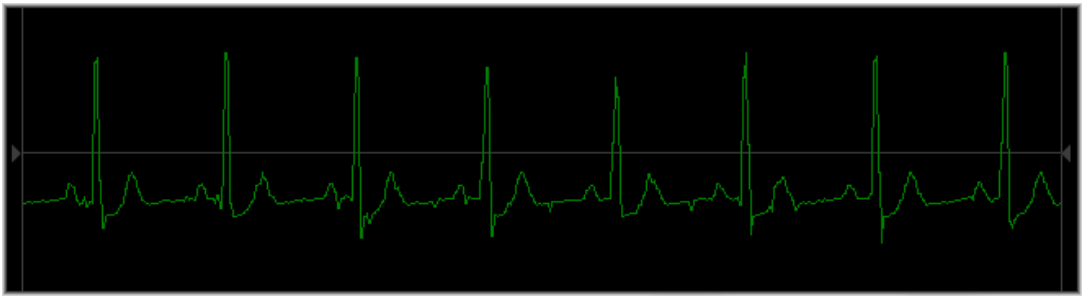
J. Η περιοχή όπου εμφανίζεται ο ασθενής και οι επιπλέον πληροφορίες, όταν είναι επιλεγμένες.

K. Εμφανίζει πάνω από τον ασθενή τις πληροφορίες σεναρίου, όπως ακριβώς είναι και στην οθόνη επιλογής σεναρίου.

L. Εδώ παρουσιάζεται κάποιο μήνυμα προς τον χρήστη, το οποίο εξαρτάται πάντα από το σενάριο.



Σχήμα 5.7: Το κουμπί Zoom δεν είναι επιλεγμένο

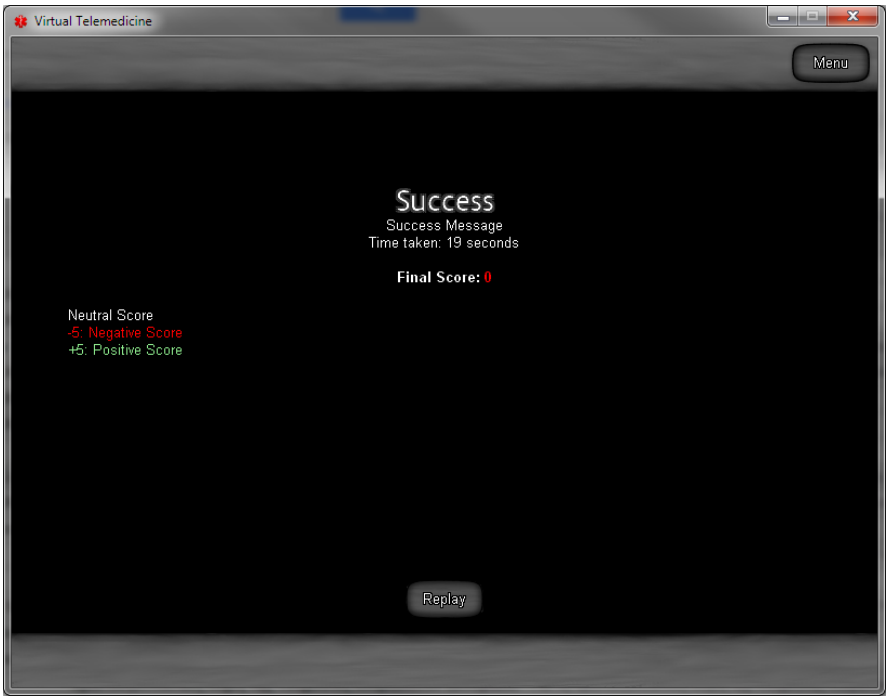


Σχήμα 5.8: Το κουμπί Zoom είναι επιλεγμένο

Σχήμα 5.9: Η βαθμολογία του χρήστη

5.4.4 Τέλος του παιχνιδιού

Όταν ένα σενάριο τερματίσει (Σχήμα 5.10), έχει δύο δυνατά αποτελέσματα: Αυτό της επιτυχίας και αυτό της αποτυχίας. Αυτές οι πληροφορίες εμφανίζονται και στην περίπτωση της αποτυχίας, με τη διαφορά ότι το μήνυμα γράφει «Failure» αντί του «Success».



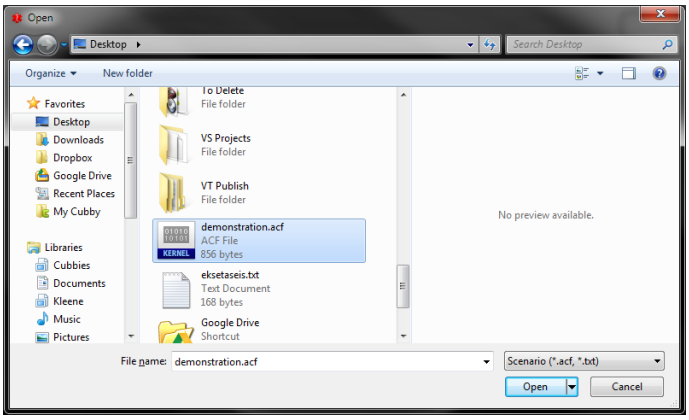
Σχήμα 5.10: Το σενάριο τερμάτισε με αποτέλεσμα την επιτυχία

Με το πάτημα του κουμπιού *Replay* το σενάριο επανεκκινεί από την αρχή, ενώ με το πάτημα του κουμπιού *Main Menu* ο χρήστης μετακινείται στο Κυρίως Μενού.

5.4.5 Εισαγωγή και επεξεργασία νέων Σεναρίων

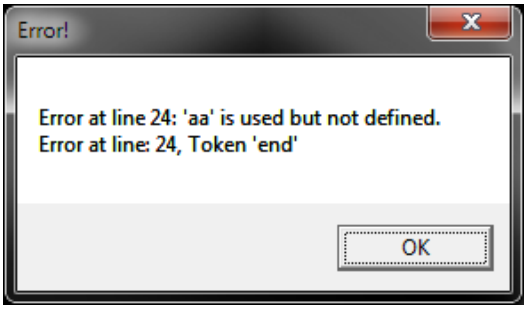
Όταν ο χρήστης επιλέξει *Scenario Manager* από το Κυρίως Μενού, μετακινείται στην οθόνη Διαχείρισης Σεναρίων (Σχήμα 5.11). Πατώντας το κουμπί *Add*, ζητείται από τον χρήστη να εξακριβώσει πού βρίσκεται το σενάριο το οποίο θέλει να προσθέσει (Σχήμα 5.12).

Σχήμα 5.11: Διαχείριση σεναρίων

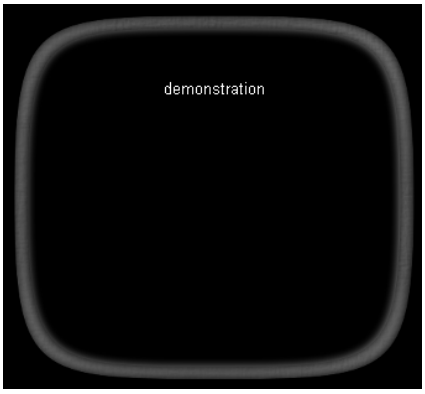


Σχήμα 5.12: Επιλογή σεναρίου από τον υπολογιστή

Στην περίπτωση που το σενάριο περιέχει συντακτικά λάθη, τότε αυτά θα εμφανιστούν σε νέο παράθυρο (Σχήμα 5.13). Αν διαφορετικά το σενάριο μεταγλωττιστεί με επιτυχία, θα εμφανιστεί στην λίστα των σεναρίων (Σχήμα 5.14).



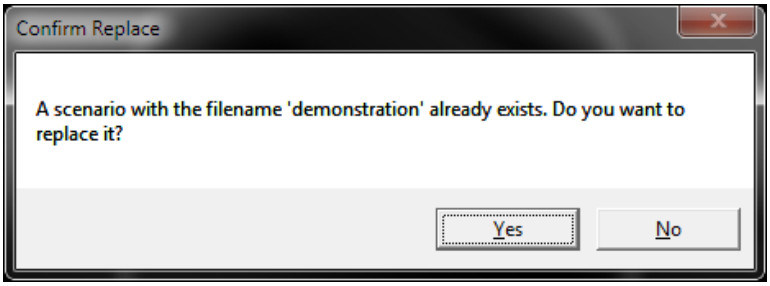
Σχήμα 5.13: Η μεταγλώττιση παρουσίασε συντακτικά λάθη



Σχήμα 5.14: Το σενάριο προστέθηκε με επιτυχία

Το όνομα που παρουσιάζεται είναι το όνομα του αρχείου και όχι του σεναρίου. Όταν ο χρήστης μεταγλωττίσει με επιτυχία ένα σενάριο που έχει όνομα αρχείου το ίδιο με το

όνομα αρχείου κάποιου υπάρχον σεναρίου, θα ερωτηθεί αν θέλει να το αντικαταστήσει (Σχήμα 5.15).



Σχήμα 5.15: Ένα σενάριο με το ίδιο όνομα αρχείου υπάρχει ήδη στη λίστα

Τέλος, με τα κουμπιά *Delete* και *Rename* ο χρήστης έχει τη δυνατότητα να διαγράψει κάποιο σενάριο που υπάρχει στη λίστα ή να αλλάξει το όνομα αρχείου του.

Τα αρχεία αυτά βρίσκονται και στο χώρο *Documents\Virtual Telemedicine\Scripts* και περιέχουν κώδικα στην Ενδιάμεση Γλώσσα και όχι στην γλώσσα σεναρίων Scribulance. Έτσι, αν ο χρήστης θελήσει, μπορεί να διαχειριστεί τα σενάρια του και έξω από το παιχνίδι.

Κεφάλαιο 6

Αξιολόγηση

6.1 Απαιτήσεις

Οι απαιτήσεις που αναφέρονται στο Κεφάλαιο 3 έχουν ικανοποιηθεί πλήρως από την υλοποίηση του Virtual Telemedicine όσον αφορά τις γενικές απαιτήσεις και τις απαιτήσεις του γραφικού περιβάλλοντος. Οι γενικές απαιτήσεις ικανοποιήθηκαν με την ανάπτυξη της γλώσσας σεναρίων Scribulance, με την οποία μπορούν να γραφτούν όλων των ειδών σενάρια που θα μπορούσαν να συμβούν. Οι γραφικές απαιτήσεις επίσης καλύφθηκαν από το γραφικό περιβάλλον.

Όσον αφορά τις ιατρικές και εκπαιδευτικές απαιτήσεις, έχουν επίσης ικανοποιηθεί με τη διαφορά ότι εξαρτάται φυσικά από το σενάριο. Αν ένα σενάριο δεν είναι γραμμένο σωστά, τότε μπορεί, για παράδειγμα, να μην βαθμολογεί τον χρήστη ή να μην παρέχει το ιστορικό του ασθενή. Θεωρούμε πως οποιαδήποτε νέα σενάρια κυκλοφορήσουν από εμάς στο μέλλον θα ικανοποιούν τις απαιτήσεις μας, όμως δεν μπορούμε να εγγυηθούμε την ορθότητα των σεναρίων που θα γράψουν τρίτοι.

6.2 Μεθοδολογία Αξιολόγησης Ευχρηστίας Παιχνιδιού

Το προτεινόμενο μοντέλο αξιολόγησης για το Virtual Telemedicine είναι η Αξιολόγηση Ευχρηστίας (Usability Testing). Η Αξιολόγηση Ευχρηστίας περιλαμβάνει τη μέτρηση των επιδόσεων τυπικών χρηστών όταν εκτελούν ειδικές προκαθορισμένες διαδικασίες, οι οποίες συμβαδίζουν μ' αυτές για τις οποίες ένα σύστημα είναι κατασκευασμένο [29]. Οι δοκιμές πραγματοποιούνται συχνά σε ελεγχόμενες συνθήκες εργαστηρίου. Όταν οι δοκιμές φθάσουν στο τέλος τους, τα αποτελέσματα καταγράφονται και αναλύονται. Ερωτηματολόγια δίνονται στους χρήστες και οι απαντήσεις τους κατηγοριοποιούνται και υπολογίζεται η μέση βαθμολογία. Το Usability Testing μπορεί να συνδυαστεί με παρατηρήσεις, βιντεοσκοπώντας τους χρήστες και καταγράφοντας τις ενέργειες τους με το λογισμικό.

6.3 Συμμετέχοντες

Οι χρήστες οι οποίοι στοχεύονται για την αξιολόγηση του Virtual Telemedicine θα είναι γιατροί. Η δειγματοληψία της μελέτης αυτής θα περιέχει περίπου 20 κατοίκους από διάφορα μέρη της Ελλάδας. Λόγω των ειδικών χρονικών περιορισμών, την περιορισμένη διαθεσιμότητα των γιατρών και τη γεωγραφική τους διαφορά, καθώς οι γιατροί ζουν και εργάζονται σε διαφορετικές πόλεις στην Ελλάδα, το Usability Testing δεν είναι εφικτό να πραγματοποιηθεί σε εργαστήριο. Λεπτομερείς βήμα-προς-βήμα οδηγίες θα σταλούν στους γιατρούς, έτσι ώστε να μπορέσουν να εγκαταστήσουν το παιχνίδι στους προσωπικούς τους υπολογιστές και να το δοκιμάσουν με την ησυχία τους. Οι συμμετέχοντες στη συνέχει ζητήθηκαν να συμπληρώσουν ένα online ερωτηματολόγιο και να στείλουν στους ερευνητές ένα αρχείο καταγραφής, το οποίο καταγράφει αυτόματα το παιχνίδι και περιγράφει τις ενέργειες τους. Το αρχείο αυτό βρίσκεται κάτω από τον φάκελο των *My Documents/Virtual Telemedicine* ως *Log.txt*.

6.4 Όργανο Αξιολόγησης

Το όργανο αξιολόγησης θα είναι ένα ερωτηματολόγιο που έχει αναπτυχθεί ειδικά για αυτό το παιχνίδι, το οποίο αποτελείται από τέσσερα μέρη. Το πρώτο μέρος αναφέρεται στα δημογραφικά στοιχεία των συμμετεχόντων εκ των οποίων είναι το φύλο, η ηλικία, τα έτη εμπειρίας στην ιατρική εκπαίδευση, περιοχή της ιατρικής τους γνώσης και ειδίκευσης και η προηγούμενη εμπειρία με παιχνίδια, σοβαρά παιχνίδια και σοβαρά παι-

χνίδια στον ιατρικό τομέα. Το δεύτερο μέρος ζητά από τους συμμετέχοντες να εγκαταστήσουν το παιχνίδι, να προσπαθήσουν να λύσουν τα προβλήματα των τριών σεναρίων και στη συνέχεια να βαθμολογήσουν το επίπεδο δυσκολίας για κάθε μία από τις ακόλουθες ενέργειες:

1. Install the game
2. Select one of the proposed medication/treatments for the patient
3. Keep track of their score while playing the game
4. Access the game objectives
5. Access the game instructions
6. Access the patient history
7. Solve the problem

Μια κλίμακα τεσσάρων σημείων Likert χρησιμοποιείται για την αξιολόγηση από τις ανωτέρω ενέργειες, με τις ακόλουθες επιλογές: α) *Easy*, β) *Ok*, γ) *Difficult*, δ) *Needed help*.

Το τρίτο μέρος του ερωτηματολογίου περιλαμβάνει 21 σημεία που εστιάζουν στην αξιολόγηση του παιχνιδιού. Οι συμμετέχοντες καλούνται να βαθμολογήσουν τις καταστάσεις που παρουσιάζονται παρακάτω χρησιμοποιώντας μια κλίμακα τεσσάρων σημείων Likert που περιλαμβάνει τις ακόλουθες επιλογές: α) *Strongly Disagree*, β) *Disagree*, γ) *Agree*, δ) *Strongly Agree* και ε) *I don't know*.

Τα στοιχεία είναι τα ακόλουθα:

1. The game will be interesting for medical students.
2. The game is useful for medical student training.
3. The game is user-friendly.
4. The game provides ways to recover after making a mistake.
5. I like the interface of the game.
6. The game graphics are adequate.
7. The terminology used is correct.
8. The terminology used is consistent.
9. The response time of the game is as expected.
10. The feedback I receive when I make a choice is adequate.

11. The feedback I receive when I make a choice is confusing.
12. I can learn from my mistakes when I play the game.
13. The game is not challenging for me.
14. I feel “in control” when I play the game.
15. The time allowed by the game for the doctor to save the patient is sufficient.
16. I needed more time to be able to solve the problem in Scenario Precordial Pain.
17. I needed more time to be able to solve the problem in Scenario Syncope.
18. I needed more time to be able to solve the problem in Scenario Palpitations.
19. If I were an instructor I would like to use the game in a classroom setting with my students.
20. I would recommend the game to my colleagues.
21. The game is complicated.

Το τέταρτο μέρος του ερωτηματολογίου αποτελείται από πέντε ανοιχτές ερωτήσεις, οι οποίες είναι οι εξής:

1. What did you like about the Virtual Telemedicine serious game?
2. What did you not like about the Virtual Telemedicine serious game?
3. What did you find confusing or difficult to use in Virtual Telemedicine?
4. How would you suggest improving Virtual Telemedicine?
5. If you were training medical students, would you be interested in using this game in your class?

6.5 Αποτελέσματα

Έχουμε ήδη λάβει 4 ολοκληρωμένα ερωτηματολόγια και αναμένουμε τα υπόλοιπα. Η ηλικία των ερωτηθέντων κυμαίνεται από 31 ως 36 με μέσο όρο το 33. Προέρχονται από 2 διαφορετικά ιατρικά πανεπιστήμια της Ελλάδας και εργάζονται σε 3 διαφορετικά νοσοκομεία. Η περιοχή ενασχόλησης τους είναι Καρδιολογία, Οφθαλμολογία, Παθολογία και Γενική, ενώ ο ιατρικός τους ρόλος διαφέρει. Μόνο ένας από τους ερωτηθέντες έχει προηγούμενη εμπειρία σε σοβαρά παιχνίδια.

Οι απαντήσεις του δεύτερου μέρους αναγράφονται στον πίνακα Πίνακας 6.1, οι απαντήσεις που λάβαμε όσον αφορά την αξιολόγηση του παιχνιδιού αναγράφονται στον πίνακα Πίνακας 6.2 και οι απαντήσεις στις ανοικτές ερωτήσεις αναγράφονται στον πίνακα Πίνακας 6.3.

1. Install the Game:

Χρήστης Α	Χρήστης Β	Χρήστης Γ	Χρήστης Δ
Easy	Easy	Ok	Easy

2. Select one of the proposed medication/treatments for the patient:

Χρήστης Α	Χρήστης Β	Χρήστης Γ	Χρήστης Δ
Easy	Ok	Ok	Ok

3. Keep track of their score while playing the game:

Χρήστης Α	Χρήστης Β	Χρήστης Γ	Χρήστης Δ
Easy	Ok	Ok	Ok

4. Access the game objectives:

Χρήστης Α	Χρήστης Β	Χρήστης Γ	Χρήστης Δ
Easy	Easy	Easy	Ok

5. Access the game instructions:

Χρήστης Α	Χρήστης Β	Χρήστης Γ	Χρήστης Δ
Easy	Easy	Easy	Ok

6. Access the patient history:

Χρήστης Α	Χρήστης Β	Χρήστης Γ	Χρήστης Δ
Easy	Easy	Easy	Ok

7. Solve the problem:

Χρήστης Α	Χρήστης Β	Χρήστης Γ	Χρήστης Δ
Easy	Ok	Ok	Ok

Πίνακας 6.1 Επίπεδο δυσκολίας

Evaluation of the Game:

1.The game will be interesting for medical students.

Χρήστης Α	Χρήστης Β	Χρήστης Γ	Χρήστης Δ
Agree	Strongly Agree	Strongly Agree	I don’t know

2. The game is useful for medical student training.

Χρήστης Α	Χρήστης Β	Χρήστης Γ	Χρήστης Δ
Agree	Strongly Agree	Strongly Agree	Strongly Agree

3. The game is user-friendly.

Χρήστης Α	Χρήστης Β	Χρήστης Γ	Χρήστης Δ
Agree	Strongly Agree	Strongly Agree	Agree

4. The game provides ways to recover after making a mistake.

Χρήστης Α	Χρήστης Β	Χρήστης Γ	Χρήστης Δ
Agree	Agree	Disagree	Agree

5. I like the interface of the game.

Χρήστης Α	Χρήστης Β	Χρήστης Γ	Χρήστης Δ
Agree	Agree	Strongly Agree	Agree

6. The game graphics are adequate.

Χρήστης Α	Χρήστης Β	Χρήστης Γ	Χρήστης Δ
Agree	Agree	Agree	Agree

7. The terminology used is correct.

Χρήστης Α	Χρήστης Β	Χρήστης Γ	Χρήστης Δ
Agree	Strongly Agree	Strongly Agree	Agree

8. The terminology used is consistent.

Χρήστης Α	Χρήστης Β	Χρήστης Γ	Χρήστης Δ
Agree	Strongly Agree	Agree	Agree

9. The response time of the game is as expected.

Χρήστης Α	Χρήστης Β	Χρήστης Γ	Χρήστης Δ
Agree	Agree	Disagree	Agree

10. The feedback I receive when I make a choice is adequate.

Χρήστης Α	Χρήστης Β	Χρήστης Γ	Χρήστης Δ
Agree	Disagree	Disagree	Agree

11. I can learn from my mistakes when I play the game.

Χρήστης Α	Χρήστης Β	Χρήστης Γ	Χρήστης Δ
Agree	Disagree	Agree	Strongly Agree

12. The game is not challenging for me.

Χρήστης Α	Χρήστης Β	Χρήστης Γ	Χρήστης Δ
Disagree	Strongly Disagree	Strongly Disagree	Disagree

13. The feedback I receive when I make a choice is confusing:

Χρήστης Α	Χρήστης Β	Χρήστης Γ	Χρήστης Δ
Agree	Disagree	Disagree	Agree

14. I feel “in control” when I play the game.

Χρήστης Α	Χρήστης Β	Χρήστης Γ	Χρήστης Δ
Agree	Agree	Strongly Agree	Agree

15. The time allowed by the game for the doctor to save the patient is sufficient.

Χρήστης Α	Χρήστης Β	Χρήστης Γ	Χρήστης Δ
Disagree	I don’t know	Strongly Agree	Agree

16. I needed more time to be able to solve the problem in Scenario Precordial Pain.

Χρήστης Α	Χρήστης Β	Χρήστης Γ	Χρήστης Δ
Agree	Disagree	Agree	Agree

17. I needed more time to be able to solve the problem in Scenario Syncope.

Χρήστης Α	Χρήστης Β	Χρήστης Γ	Χρήστης Δ
Agree	Disagree	Agree	Disagree

18. I needed more time to be able to solve the problem in Scenario Palpitations.

Χρήστης Α	Χρήστης Β	Χρήστης Γ	Χρήστης Δ
Agree	Agree	Strongly Agree	Disagree

19. If I were an instructor I would like to use the game in a classroom setting with my students.

Χρήστης Α	Χρήστης Β	Χρήστης Γ	Χρήστης Δ
Agree	Strongly Agree	Agree	Strongly Agree

20. I would recommend the game to my colleagues.

Χρήστης Α	Χρήστης Β	Χρήστης Γ	Χρήστης Δ
Agree	Strongly Agree	Strongly Agree	Strongly Agree

21. The game is complicated.

Χρήστης Α	Χρήστης Β	Χρήστης Γ	Χρήστης Δ
Disagree	Disagree	Strongly Agree	Strongly Disagree

Πίνακας 6.2 Αξιολόγηση του Παιχνιδιού

Questions

1. What did you like about the Virtual Telemedicine serious game?

Χρήστης Α	It was interesting…
Χρήστης Β	It is simple, not many things on the screen to confuse the user.
Χρήστης Γ	It was ok
Χρήστης Δ	Is a meaningful game on the cardiology even for those who are not to direct the object.

2. What did you not like about the Virtual Telemedicine serious game?

Χρήστης Α	Everything was very good… Congratulations
Χρήστης Β	I believe there is a space for more feedback on the wrong answers, either during the game or after it ends (because the patient comes first, you have to save gun and then study…)
Χρήστης Γ	Nothing
Χρήστης Δ	Few scenarios. History with the same potential alternative – different responses

3. What did you find confusing or difficult to use in Virtual Telemedicine?

Χρήστης Α	Everything was easy…
Χρήστης Β	I did not find anything confusing in the game.
Χρήστης Γ	Nothing
Χρήστης Δ	Nothing

4. How would you suggest improving Virtual Telemedicine?

Χρήστης Α	I cannot suggest something…

Χρήστης Β	I think it is a little bit more concise than it should; after all it is an educational game, so I think a little bit more information/ feedback should be given on the answers.
Χρήστης Γ	No
Χρήστης Δ	Better Graphics More scenarios History with the same potential alternative – different responses

5. If you were training medical students, would you be interested in using this game in your class?

Χρήστης Α	Yes I would be interested in using this game in my class...
Χρήστης Β	Yes, I think it would be interesting for the students.
Χρήστης Γ	Yes
Χρήστης Δ	Yes

Πίνακας 6.3 Ανοιχτές Ερωτήσεις

Το παιχνίδι εγγράφει αυτόματα τις ενέργειες του χρήστη και τις τοποθετεί σε ένα αρχείο που βρίσκεται στην τοποθεσία *(user)\My Documents\Virtual Telemedicine\Log.txt*. Δύο από τους χρήστες που δοκίμασαν το παιχνίδι μας έχουν στείλει τα αρχεία αυτά για σκοπούς αξιολόγησης. Τα περιεχόμενα των αρχείων αυτών βρίσκονται στους Πίνακας 6.4 και Πίνακας 6.5.

```
----------
Time: 29/10/2012 7:05 μμ
Scenario: Syncope
Result: Failure
Score: 35

-10: You have wrongfully chosen 'First degree atrioventricular block'
+15: You have correctly chosen 'Complete atrioventricular block'
+15: You have correctly chosen 'Ensure venous line and provide fluids'.
-10: You have wrongfully provided 'Verapamil'
----------

----------
Time: 29/10/2012 7:06 μμ
Scenario: Syncope
Result: Success
Score: 85

+15: You have correctly chosen 'Complete atrioventricular block'
+15: You have correctly chosen 'Ensure venous line and provide fluids'.
+15: You have correctly provided 'Atropine'
+15: You have correctly provided 'Pacing'!
----------

----------
Time: 29/10/2012 7:12 μμ
Scenario: Palpitations
Result: Success
```

Score: 75

+15: 'Atrial fibrillation with rapid ventricular response' was the correct action given the circumstances.
+15: You have correctly provided 'Amiodarone'.
+10: You have correctly provided 'Heparin LMWH'.

Time: 29/10/2012 7:16 μμ
Scenario: Precordial Pain
Result: Success: 20
Score: 91

+10: You have provided Nitroglycerin sublingual
+5: You have provided 4mg of Morphine (Total: 4mg)
+5: You have provided 100mg of Aspirine
+5: You have provided 4lt of Oxygen (Total: 4lt)
+10: You have provided 5mg of Metoprolol (Total: 5mg)
-1: Not enough power
+1: 360J Defibrillation
-5: Too much power
+1: 360J Defibrillation
+10: The patient is restored!
+10: You have correctly provided Clopidogrel
+15: The patient is saved!

Πίνακας 6.4 Περιεχόμενα αρχείου εγγραφής από τον Χρήστη Β

Time: 31/10/2012 8:48 μμ
Scenario: Syncope
Result: Failure
Score: 45

-10: You have wrongfully chosen 'Gulf-ventricular block Mobitz type 1 (Weckenbach)'
-10: You have wrongfully chosen 'First degree atrioventricular block'
+15: You have correctly chosen 'Complete atrioventricular block'
-10: You have wrongfully chosen 'Pacemaker'
-10: You have wrongfully chosen 'Lidocaine'
+15: You have correctly chosen 'Ensure venous line and provide fluids'.
+15: You have correctly provided 'Atropine'
+15: You have correctly provided 'Pacing'!

Time: 31/10/2012 8:48 μμ
Scenario: Syncope
Result: Success
Score: 85

+15: You have correctly chosen 'Complete atrioventricular block'
+15: You have correctly chosen 'Ensure venous line and provide fluids'.
+15: You have correctly provided 'Atropine'
+15: You have correctly provided 'Pacing'!

Time: 31/10/2012 8:53 μμ

Scenario: Palpitations
Result: Success
Score: 65

-10: 'Atrial fibrillation with ventricular response good' was an incorrect action given the circumstances.
+15: 'Atrial fibrillation with rapid ventricular response' was the correct action given the circumstances.
+15: You have correctly provided 'Amiodarone'.
+10: You have correctly provided 'Heparin LMWH'.

Time: 31/10/2012 8:56 μμ
Scenario: Precordial Pain
Result: Failure: 22
Score: 12

+5: You have provided 4lt of Oxygen (Total: 4lt)
+10: You have provided Nitroglycerin sublingual
+5: You have provided 4mg of Morphine (Total: 4mg)
+5: You have provided 100mg of Aspirine
-20: You have wrongfully provided Codeine
+10: You have provided 5mg of Metoprolol (Total: 5mg)
+1: 360J Defibrillation
-5: Too much power
+2: 200J Defibrillation
-1: Not enough power
+10: The patient is restored!
-15: Alterplase was a bad decision
-20: The patient is dead!

Time: 31/10/2012 8:57 μμ
Scenario: Precordial Pain
Result: Success: 20
Score: 97

+5: You have provided 4lt of Oxygen (Total: 4lt)
+10: You have provided 5mg of Metoprolol (Total: 5mg)
+10: You have provided Nitroglycerin sublingual
+5: You have provided 4mg of Morphine (Total: 4mg)
+5: You have provided 100mg of Aspirine
+1: 360J Defibrillation
-1: Not enough power
+2: 200J Defibrillation
+10: The patient is restored!
+10: You have correctly provided Clopidogrel
+15: The patient is saved!

Πίνακας 6.5 Περιεχόμενα αρχείου εγγραφής από τον Χρήστη Δ

Κεφάλαιο 7

Συμπεράσματα και Μελλοντική Εργασία

7.1 Συμπεράσματα

Αυτή η διπλωματική εργασία περιγράφει την υλοποίηση ενός σοβαρού παιχνιδιού, το Virtual Telemedicine, το οποίο στοχεύει να χρησιμοποιηθεί ως σημαντικό εργαλείο για την εκπαίδευση γιατρών σε χρήση τεχνολογιών τηλεϊατρικής.

Το παιχνίδι δημοσιεύτηκε σε δύο συνέδρια μετά από κρίση ([2] και [3]). Απόσπασε καλές κριτικές και το ενδιαφέρον από διάφορους γιατρούς που επιθυμούν τη συνεργασία μαζί μας. Έχει προσκληθεί να συμπεριληφθεί ως δημοσίευση στο Journal of Medical Internet Research. Η αξιολόγηση του παιχνιδιού βρίσκεται υπό εξέλιξη και αναμένονται οι απαντήσεις στο ερωτηματολόγιο που αναφέραμε πιο πάνω.

Για τους σκοπούς του παιχνιδιού αναπτύχθηκε και η πρώτη έκδοση της γλώσσας σεναρίων Scribulance, η οποία είναι κατάλληλη για συγγραφή σεναρίων βασισμένα σε καταστάσεις (state-based scenarios). Είναι επαναχρησιμοποιήσιμη και επεκτάσιμη, χρησιμοποιώντας το σύστημα συναρτήσεων επέκτασης, και μπορεί να χρησιμοποιηθεί και για άλλα έργα. Στην παρούσα έκδοση ο εκτελεστής των σεναρίων μπορεί να χρησιμοποιηθεί ως βιβλιοθήκη της .NET.

Για την ικανοποίηση των απαιτήσεων του παιχνιδιού αναπτύχθηκε και ένας επεξεργαστής των σημάτων ECG, ο οποίος έχει τη δυνατότητα να δημιουργήσει ή τροποποιήσει

σήματα ECG και να παράγει ως αποτέλεσμα σήματα ειδικά για κάποιες παθήσεις για τις οποίες είναι δύσκολο να ανακτηθούν από πραγματικούς ασθενείς.

7.2 Συμπεράσματα Αποτελεσμάτων

Οποιαδήποτε αποτελέσματα έχουμε για το παιχνίδι είναι, δυστυχώς, προκαταρκτικά λόγω της μικρής βάσης απαντήσεων και άρα το τελικό συμπέρασμα μπορεί να διαφέρει. Παρατηρούμε όμως μια γενικώς πολύ θετική αποδοχή του παιχνιδιού, κυρίως όσον αφορά τον εκπαιδευτικό τομέα ο οποίος μας ενδιαφέρει περισσότερο. Οποιαδήποτε κριτική είναι εποικοδομητική και όχι αποθαρρυντική.

Οι χρήστες είναι ευχαριστημένοι με την ιδέα και τους μηχανισμούς του παιχνιδιού. Η διαπροσωπεία είναι απλή και ταυτόχρονα πληροφοριακή. Η χρήση του παιχνιδιού είναι απλή και εύκολη, χωρίς να ταλαιπωρεί τους χρήστες. Οποιεσδήποτε πληροφορίες οι οποίες είναι διαθέσιμες από το παιχνίδι είναι εύκολα προσβάσιμες από τους χρήστες και δε χρειάζεται να σπαταλήσουν χρόνο να ψάξουν γι’ αυτές. Όλοι οι χρήστες μέχρι τώρα συμφώνησαν πως το παιχνίδι μπορεί να χρησιμοποιηθεί ως ένα αποτελεσματικό εργαλείο για εκπαίδευση στον ιατρικό τομέα.

Τα αρνητικά σχόλια μέχρι τώρα αφορούσαν περισσότερο τα σενάρια του παιχνιδιού και όχι το ίδιο το παιχνίδι, του οποίου η επιτυχία φαίνεται να είναι ομόφωνα αποδεκτή . Αυτό σημαίνει πως ίσως χρειαστεί να προστεθούν περισσότερες λεπτομέρειες σ’ αυτά όπως για παράδειγμα περισσότερα ενδεχόμενα, ειδικά όταν ο χρήστης κάνει λάθος. Εδώ όμως φανερώνεται και η χρησιμότητα της γλώσσας σεναρίων Scribulance: μπορεί ο οποιοσδήποτε γιατρός να δημιουργήσει νέο σενάριο, ή ακόμη να βελτιώσει / συμπληρώσει ένα υπάρχον σενάριο του οποίου ο πηγαίος κώδικας είναι διαθέσιμος.

7.3 Μελλοντική Εργασία

7.3.1 Εκπαιδευτικό Μέρος και Αξιολόγηση

Ένα από τα επόμενα βήματα αυτού του σχεδίου αφορά την εφαρμογή του Virtual Telemedicine στη χρήση της προσομοίωσης στο πλαίσιο για το οποίο προοριζόταν. Αυ-

τό θα επιτρέψει την εξέταση του κατά πόσο οι μαθητές μαθαίνουν πραγματικά τις έννοιες που αποτελούν μέρος της προσομοίωσης και αν είναι πράγματι σε θέση να λύσουν τα προβλήματα όσον αφορά τις ιατρικές καταστάσεις έκτακτης ανάγκης. Αυτό είναι σύμφωνα με τις κατευθυντήριες γραμμές που παρέχονται από Graafland et al. (2012) ο οποίος τόνισε ότι τα παιχνίδια πρέπει να είναι σχεδιασμένα ώστε να συμβαδίζουν με προγράμματα διδασκαλίας αν πρόκειται να χρησιμοποιηθούν ως ένας τρόπος για την πρόληψη των ιατρικών σφαλμάτων [30].

Χωρίς πραγματική εμπειρία, είναι δυνατό για τους μαθητές να είναι επιτυχείς στην προσομοίωση, χωρίς κατ' ανάγκη να αποκτήσουν την ικανότητα να εφαρμόσουν σωστά τις ίδιες αρχές σε άλλες περιπτώσεις. Η συμπεριφορά τους εντός της προσομοίωσης μπορεί να είναι δοκιμαστική ή να βασίζεται σε λανθασμένη, αν και επιτυχημένη, λογική. Κατά τη διάρκεια μιας συνεδρίας απολογισμού που θα σχεδιαστεί όταν το Virtual Telemedicine χρησιμοποιηθεί σε ένα εκπαιδευτικό πλαίσιο, ως μέρος του σχεδίου εφαρμογής του, οι εκπαιδευόμενοι θα πρέπει να εξηγήσουν την κατανόησή τους για την τρέχουσα κατάσταση, την επιθυμητή κατάσταση, το σύνολο των αρχών που εφαρμόστηκαν για την επίλυση του προβλήματος και πώς ακριβώς οι ενέργειές τους μετέτρεψαν την τρέχουσα κατάσταση στην επιθυμητή.

Όσον αφορά την τελική αξιολόγηση η επόμενη μεταβλητή που θα μετρηθεί είναι η μεταφορά, η εφαρμογή της νέας μάθησης και σε άλλες καταστάσεις, η οποία στο πλαίσιο της παρούσας ερευνητικής μελέτης θα μετρήσει τη «σχετική μεταφορά», όπως την ικανότητα για την επίλυση παρόμοιων προβλημάτων σε παρόμοιες καταστάσεις.

Όσον αφορά τη μελλοντική έρευνα, όπως ο Graafland et al. (2012) επεσήμανε, τα παιχνίδια που αναπτύσσονται ή χρησιμοποιούνται για την εκπαίδευση του ιατρικού προσωπικού θα πρέπει να επιθεωρηθούν πριν από την ένταξή τους στις μεθόδους διδασκαλίας. Ως εκ τούτου μια περαιτέρω έρευνα θα πρέπει να καθορίσει τις παραμέτρους και να επιβεβαιώσει επίσημα για το αν ένα σοβαρό παιχνίδι μπορεί να θεωρηθεί ως πλήρες μέσο διδασκαλίας για ιατρικούς και χειρουργικούς επαγγελματίες [30].

7.3.2 Τεχνικό Μέρος

Όσον αφορά το τεχνικό κομμάτι, μέρος του κώδικα θα ήταν καλό να ξαναγραφτεί έτσι ώστε το παιχνίδι και η γλώσσα Scribulance να μπορούν να χρησιμοποιηθούν σε περισσότερα μηχανήματα διαφόρων λειτουργικών συστημάτων. Το κομμάτι του εκτελεστή θα βελτιστοποιηθεί και θα ενσωματωθεί εν μέρει με το κομμάτι του μεταγλωττιστή, έτσι ώστε να αποκρύβεται εντελώς το ενδιάμεσο μέρος και χωρίς να χρειάζεται πλήρης μεταγλώττιση για την ανάκτηση των Πληροφοριών Σεναρίου. Η Scribulance επίσης θα επεκταθεί, επιτρέποντας καλύτερη οργάνωση πολύπλοκων σεναρίων.

Όσον αφορά τη διαχείριση των σεναρίων, θα αναπτυχθεί σύστημα το οποίο θα κατεβάζει αυτόματα νέα σενάρια τα οποία είναι εγκεκριμένα από γιατρούς και πληρούν τις εκπαιδευτικές και ιατρικές απαιτήσεις του παιχνιδιού. Το παιχνίδι θα προειδοποιεί τον χρήστη όταν δοκιμάσει να παίξει σενάριο από τρίτους, το οποίο δεν περάσει από τον έλεγχο των αρμόδιων ατόμων, ούτως ώστε να ξέρει πως το σενάριο ενδέχεται να περιέχει λάθη.

Βιβλιογραφία

[1] E. Kyriacou, S. Pavlopoulos, A. Berler, M. Neophytou, A. Bourka, A. Georgoulas, A. Anagnostaki, D. Karayiannis, C. Schizas, C. Pattichis, A. Andreou, and D. Koutsouris, "Multi-purpose HealthCare Telemedicine Systems with mobile communication link support," *BioMedical Engineering OnLine*, vol. 2, no. 1, p. 7, Mar. 2003.

[2] I. Nicolaidou, A. Antoniades, C. Marangos, E. Dafli, K. Efthyvoulos, P. Bamidis, and C. Pattichis, "The Instructional Design of Virtual Telemedicine: A Simulation-Based Serious Game in Health-Care," *ICERI2012 Proceedings*, pp. 5746–5755, 2012.

[3] A. Antoniades, C. Marangos, K. Efthyvoulos, I. Nicolaidou, E. Dafli, P. Bamidis, and C. Pattichis, "Virtual Telemedicine: A Serious Game to Develop Problem Solving Skills in Medical Education," *EDULEARN12 Proceedings*, pp. 3612–3619, 2012.

[4] S. Pavlopoulos, E. Kyriacou, A. Berler, S. Dembeyiotis, and D. Koutsouris, "A novel emergency telemedicine system based on wireless communication technology-AMBULANCE," *IEEE Transactions on Information Technology in Biomedicine*, vol. 2, no. 4, pp. 261 –267, Dec. 1998.

[5] K. Salen and E. Zimmerman, "Rules of Play: Game Design Fundamentals.," MIT Press, 2003, p. 80.

[6] "Electronic game - Wikipedia, the free encyclopedia." [Online]. Available: http://en.wikipedia.org/wiki/Electronic_game. [Accessed: 18-Dec-2012].

[7] M. Ulisack and M. Wright, "Games in education: Serious Games.," *Bristol, Futurelab*, 2010.

[8] A. J. Stapleton, "Serious games: Serious opportunities.," presented at the Australian Game Developers' Conference, Academic Summit, Melbourne, 2004.

[9] “Serious Games: Incorporating Video Games in the Classroom (EDUCAUSE Quarterly) | EDUCAUSE.edu.” [Online]. Available: http://www.educause.edu/ero/article/serious-games-incorporating-video-games-classroom. [Accessed: 18-Dec-2012].

[10] M. Prensky, “True believers: Digital game-based learning in the military.,” *Digital game-based learning*, 2001.

[11] “IEEE Xplore - Games soldiers play.” [Online]. Available: http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=988702&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D988702. [Accessed: 08-Jan-2013].

[12] A. C. Hoggatt, “An experimental business game,” *Behavioral Science*, vol. 4, no. 3, pp. 192–203, 1959.

[13] “What is the advergame?The definition of advergame.” [Online]. Available: http://www.frontnetwork.net/advergame/. [Accessed: 18-Dec-2012].

[14] M. Macedonia, “Games, Simulation, and the Military Education Dilemma.” 01-Jan-2002.

[15] “History of Computing: Video games - Golden Age.” [Online]. Available: http://www.thocp.net/software/games/golden_age.htm. [Accessed: 15-Jan-2013].

[16] “Microsoft Makes Big Cuts At Flight Sim Studio.” [Online]. Available: http://www.gamasutra.com/php-bin/news_index.php?story=21981#.UOtqfG9miSp. [Accessed: 08-Jan-2013].

[17] “Information about Microsoft Flight Simulator.” [Online]. Available: http://www.bruceair.com/msfs/msfs.htm. [Accessed: 08-Jan-2013].

[18] B. Williams, *Microsoft Flight Simulator as a Training Aid: A Guide for Pilots, Instructors, and Virtual Aviators*. Aviation Supplies & Academics, Inc., 2007.

[19] S. Oss, “Computers with Wings: Flight Simulation and Personalized Landscapes,” *Journal of Science Education and Technology*, vol. 14, no. 1, pp. 117–122, Mar. 2005.

[20] S. De Freitas, “Learning in immersive worlds. A review of game-based learning.” 2006.

[21] G. Albright, R. Goldman, K. M. Shockley, F. McDevitt, and S. Akabas, “Using an Avatar-Based Simulation to Train Families to Motivate Veterans with Post-Deployment Stress to Seek Help,” *VA. Games for Health*, pp. 21–28, 2012.

[22] L. Enochsson, B. Isaksson, R. Tour, A. Kjellin, L. Hedman, T. Wredmark, and L. Tsai-Felländer, “Visuospatial skills and computer game experience influence the performance of virtual endoscopy.,” *Journal of Gastrointestinal Surgery*, pp. 874–880, 2004.

[23] S. Arnab, I. Dunwell, and K. Debattista, *Serious games for healthcare: Applications and Implications*. IGI Global, 2012.

[24] “Gamebryo | Industry-Leading Cross-Platform Game Development Engine.” [Online]. Available: http://64.239.241.166/en/Clients--Titles/Visual-Simulation-and-Serious-Games/Pulse/. [Accessed: 15-Jan-2013].

[25] “Fishing Cactus development blog » Blog Archive » Fishing Cactus presents R.O.G.E.R, the first Medical Kinect Serious game.” [Online]. Available: http://blog.fishingcactus.com/index.php/2010/10/07/fishing-cactus-presents-r-o-g-e-r-the-first-medical-kinect-serious-game/. [Accessed: 08-Jan-2013].

[26] “The Nobel Prize in Physiology or Medicine 1930.” [Online]. Available: http://www.nobelprize.org/nobel_prizes/medicine/laureates/1930/. [Accessed: 09-Jan-2013].

[27] “The Blood Typing Game - about blood groups and blood transfusions.” [Online]. Available: http://www.nobelprize.org/educational/medicine/bloodtypinggame/about.html. [Accessed: 09-Jan-2013].

[28] “The Blood Typing Game.” [Online]. Available: http://www.nobelprize.org/educational/medicine/bloodtypinggame/game/index.html. [Accessed: 25-Jan-2013].

[29] Y. Rogers, H. Sharp, and J. Preece, *Interaction design, beyond human-computer interaction*. John Wiley & Sons, 2011.

[30] M. Graafland, M. P. Schraagen, and Schijven, “Systematic review of serious games for medical education and surgical skills training,” *British Journal of Surgery*, vol. 99, 2012.

Παράρτημα Α

Scribulance for Virtual Telemedicine: Reference and Manual

UNIVERSITY OF CYPRUS

Scribulance for Virtual Telemedicine

Reference and Manual

Charis Marangos

12/20/2012

Introduction

Scribulance is a simple scripting language developed for Virtual Telemedicine, a Serious Game created by the University of Cyprus. The purpose of this scripting language is to create Scenarios for this game, which can be used for simulating real-life cardiological incidents, for educational purposes.

In order to understand what Scribulance is about, one needs to understand how the game is played.

Every scenario in the game involves some kind of incident that has befallen the patient. The person who plays the game acts as a doctor and he is called to treat the patient so that his condition is stabilized. The game is won when the patient is restored to a healthy condition, and it is lost when the patient dies.

At any given time, the doctor has to make a choice from those shown on the screen. The patient is also displayed, changing his animation or color to depict pain or any effect the scenario deems necessary. The doctor is also allowed to view the patient's medical history, in case he needs extra information. The patient's ECG is displayed above, and additional information is shown on the right, like his heart rate or temperature.

Clicking one of those choices may have an impact. For example, the patient's ECG may change, or a message can be displayed to the doctor. Some choices involve some kind of risk; they can be fatal, but also have a chance to succeed. Finally, something may happen after a given time, and the patient might die if the doctor does not react fast enough!

This is where Scribulance comes in. It is a language that can describe a scenario in such a way that the choices and their effects are well-defined by the author. The Scenario can then be added into the game easily. If there are no errors in the scenario, it will be added successfully and made available to play.

Requirements:

A simple text editor will be enough to write a scenario. Don't use a word processor like Microsoft Word! Word processors add extra unpredictable information which confuse the game and make it unable to process the scenario. A suitable program for the job is Notepad, which can be found on any Windows machine.

Having a background in basic programming is encouraged and will certainly help, but it is not required.

Basic structure of a Scenario

A Scenario is made up of States, Functions and Variables:

A State may have any of the following Events:

- A collection of Options that the player will see. An Option has a set of commands to be followed whenever that Option is selected.
- An Enter event. This is a set of commands to be followed whenever that State is activated.
- An Exit event. This is a set of commands to be followed whenever that State is deactivated.
- A collection of Time Events. For each Time Event you must specify a time in minutes and seconds. A time event is a set of commands to be followed when the specified time passes after the State has been activated.

Variables hold temporary values needed for a scenario. A variable can be, for example, the amount of times a patient was given a certain medication.

A Function is a set of commands not associated to a State. It can have any number of Parameters (also known as Arguments) and may return a single value. Functions are typically used for code that is repeated multiple times in a Scenario. Rather than copying and pasting the same code over and over and have it clutter up the scenario, you could use a function and write the code only once.

An Annotation is information, in text form, for one of the following:

- History
- Objectives
- Instructions

Moreover, for each Scenario you must specify a Description and a starting State. This is called the Scenario Declaration. This means that a Scenario must have at least one State. However, everything else is optional. A Scenario will be valid, albeit pointless, even with no Functions or Variables and just one State that has no Events.

Note: A State, a Function and an Event may also have their own Variables. These Variables are not visible anywhere else except where they were declared. Variables declared as Scenario Variables can be used anywhere in the Scenario.

The syntax of a Scenario implementing all of the above is as follows:

```
scenario "Description" StartState;


History = "History";
```

Scenario Declaration. Description is how it will appear in the scenario selection menu.

Declare the history of the patient. '=' is optional.

```
Objectives "Objectives";
Instructions "Instructions";

var SomeValue = 25;
var PI = 3.14;
var SomeOtherValue;
var SomeText = "Hello";
var LogicalValue = true;
var LogicalValue2 = false;

state StartState
begin
   var StateVariable1 = 123;

   on enter do
   begin
      var eventVar;
      //set of commands
   end

   when 0:10 do
   begin
      //set of commands
   end

   when 1:30 do
   begin
      /*
      set of
       commands
      */
   end

   option "Click me" do
   begin
      //set of commands
   end

   option "Click me instead" do
   begin
      /* set of
       commands
      */
   end

   on exit do
   begin
      //set of commands
   end
end

function CallMe()
begin
   var funcVariable;
   //set of commands
end

function CallMe2(param1, param2)
begin
   //set of commands
end

function AnotherFunction(a, b, c)
begin
   //set of commands
end
```

```
Declare the objectives of the scenario.
Declare the instructions of the scenario.

This Variable holds a number
This Variable holds a number with a decimal part
This Variable does not currently have a value
This Variable holds some text
This Variable holds the logical value true
This Variable holds the logical value false

State declaration

This Variable is only visible to this State

State Enter Event

Declare a Variable visible only to this event.


Time Event to be executed after 10 seconds



Time Event to be executed after 1 minute and 30 seconds



Option Event


Another Option Event


Exit event


State Declaration ends here

Function without parameters

Declare a Variable only visible to this Function


Function with two parameters


Function with three parameters
```

First, note the bolded words. These are called Reserved Words. They cannot be used to name States, Functions or Variables. The reserved words are the following:

and, **begin**, **call**, **div**, **do**, **else**, **end**, **function**, **goto**, **if**, **jumpto**, **mod**, **not**, **null, on**, **or**, **return**, **scenario**, **state**, **then**, **var**, **when**.

The keywords **begin** and **end** simply denote the beginning and ending of something. You may alternatively use the curly brackets **{** and **}** respectively to achieve the exact same result.

The words History, Instruction and Objectives are Annotations. Annotations can have their value set only before declaring Variables. These three annotations are optional, and can be given in any order. The '=' sign is optional, but only for Annotations; it is not optional for setting a value to a Variable.

States, Functions or Variables must be named following these guidelines:

- It must begin with a letter of the Latin alphabet or the underscore character (_)
- It is optionally followed by any series of letters or numbers, or the underscore character (_)
- It must not be a Reserved Word

Examples:

92name	invalid – starts with a number
n@me	invalid – non-latin character
name_	valid
_name	valid
name92	valid
name#2	invalid – non-latin character
long_name	valid

Note: You may not have two Functions, two States or two Variables that share the same name.

Note: Keep in mind that there exist a number of built-in Functions and Variables already defined to provide for additional functionality. These may not be overridden by the author of a Scenario. We will visit them later.

Note: Names are case-insensitive. This means that a Function called DoSomething is the same as a Function called dosomething.

The order of elements in a Scenario is as follows:

1. Scenario Declaration
2. Variable Declaration
3. State and Function declarations, in an unspecified order.

The order of events in a State is as follows:

1. Variable Declaration
2. Event Declarations, in an unspecified order. Keep in mind that the options will appear to the user in the same order that appear in the scenario.

Now note the semicolons (;) after some lines. Those lines specify a command. The Scenario Declaration and Variable Declarations are commands and so they must be succeeded with the semicolon.

Now note the lines beginning with //. These lines are comment lines and they are ignored by the computer. They're there to help other people reading your scenario to understand what it is about, but also as notes to yourself.

Finally, note the text included between /* and */. These are also comments, but they have the ability to span across multiple lines!

If you import and run this Scenario, you will be presented with the choices "Click me" and "Click me instead". Clicking them won't do anything, simply because we didn't tell it to do anything.

Values and Operators

Values are useful to a Scenario. Without them the Scenario has no point. What variables do, essentially, is that they hold a value that can be used by the Scenario.

A value can be:

- A **Text** value, also called a **String** value. A Text value must always be between double quotation marks (**"**), as seen above. They may only contain Latin characters, cannot contain more double quotes (e.g. "This "text"" is invalid, but "This text" is valid), and may span across multiple lines.

- An **Integer** value, which is a number without a decimal part.

- A **Real** value, which is a number with a decimal part. The decimal part is denoted with a dot (**.**), as seen above. A comma (**,**) is not valid. A Real value may not contain more than one decimal part.

- A **Logical** value, also called a **Boolean** value, which contains either **true** or **false**.

By using Operators, we combine Values and get a new Value. The Operators supported by Scribulance are the following:

Binary Operators:

+ Add two Numbers, two Strings, or a Number and a String together. Warning: 2 + 2 is 4, but "2" + "2" (two strings which contain a number) is "22"!

- The subtraction of two Numbers. 5.5 – 2 will return 3.5

/ Divide two numbers

***** Multiply two numbers

% The modulus of a division. 5 % 2 is 1, 8 % 3 is 2. Alternatively, you may use the word **mod** instead of %

**** The integer result of a division. 5 \ 2 is 2, 8 \ 3 is 2 and 12 \ 7 is 1. Alternatively, you may use the word **div** instead of \

^ Raise a number to a power. 2^2 is 4, 5^3 is 125

= Only applicable if you wish to assign a value to a Variable. For example, test = 2 assigns the number 2 to the Variable test.

Unary Operators:

- Negates a number

not Inverts a logical value. **true** becomes **false** and **false** becomes **true**. Alternately, you may use the exclamation mark (**!**). For example, **not true** is **false**, and **!false** is **true**.

The Logical Operators combine logical values and return either **true** or **false**.

Logical Operators:

== Whether two values are equal. You may not use a single equals sign (=) for comparison.

2 == 2 is **true**.
2 == 3 is **false**.
"word" == "word" is **true**.
"WORD" == "word" is **false**. Text comparison is sensitive to capitalization!
"24" == 24 is **false**. "24" is Text but 24 is a number!
true == true is **true**.
true == false is **false**.
true == "true" is **false**. true is a logical value but "true" is a text value!

!= Does the opposite of ==. When == is false, != is true and vice-versa. You may also use the symbols <> if you prefer.

>, >=, <, <= are used for comparing numbers with numbers or text with text.

> **Greater than**

2 **>** 1 is true.
1 **>** 2 is false.
2 > 2 is false.
"B" **>** "A" is true.
"A" **>** "B" is false.

< **Less than**

2 **<** 1 is false.
1 **<** 2 is true.
2 < 2 is false.
"B" **<** "A" is false.
"A" **<** "B" is true.

>= **Greater or equal than**

2 >= 1 is true.
1 >= 2 is false.
2 >= 2 is true.
"B" **>=** "A" is true.
"A" **>=** "B" is false.

<= **Less or equal than**

2 <= 1 is false.
1 <= 2 is true.
2 <= 2 is true.
"B" **<=** "A" is false.
"A" **<=** "B" is true.

and This returns true when both operands are true. Alternatively, you may use the symbols **&&.**

true **and** true is true.
true **and** false is false.
false **and** true is false.
false **and** false is false!
1==1 **&&** 4 > 2 is true.
1!=1 **&&** 5>2 is false.

1!=1 **and** 2>5 is false.

or This returns true when either one of the operands is true. Alternatively, you may use the symbols **||.**
true **or** true is true.
true **or** false is true.
false **or** true is true.
false **or** false is false.
1==1**||** 4 > 2 is true.
1!=1 **||** 5>2 is true.
1!=1 **or** 2>5 is false.

Since Variables hold values, they may be used in other expressions just like ordinary values would be used. For example, consider the following:

```
…

var Number = 25;
var FirstTime = false;
var HelloText = "Hello";
var SomeValue;

…

SomeValue = HelloText + Number;
SomeValue = FirstTime or true;
SomeValue = not SomeValue;

…
```

```
This Variable contains a Number value
This Variable contains a Logical value
This Variable holds a Text value
This Variable currently holds no value

The following may be placed in an event or function

SomeValue now contains "Hello25"
SomeValue now contains true, because false or true is true
SomeValue now contains false, because not true is false
```

Note: Variables, as their name suggests, might change their value any time during running the Scenario.

Note: You may not combine values to describe options or event times. Moreover, you may not use real values to describe time events. For example, the following code is erroneous:

```
…

option "Hello" + "World" do

…

when 0:(2+4) do

…
when 0:4.2 do

…
```

```
In a State:

Not allowed!

In another State:

Not allowed!

In another State:

Not allowed!
```

Finally, there are four important details to keep in mind.

- Empty values (for example, Variables with no initial value) are equal to the special value **null**. Two empty values are equal to one another.

```
…
```

```
var Variable1 = ;
var Variable2 = null;
var Test;

…

Test = Variable1 == Variable2;

…
```

```
This Variable currently holds no value
This Variable currently holds no value
This Variable currently holds no value

The following may be placed in an event or function

Test now contains true!
```

- Using operators incorrectly, the result is always **null**.

```
…

var Variable1 = "Hello";
var Variable2 = 25;
var Test = true;

…

Test = Variable1 * Variable2;

…
```

```
This Variable holds the text value "Hello"
This Variable holds the number 25
This Variable holds the logical value true

The following may be placed in an event or function

Test now contains null, because multiplying text with a number is
a valid operation!
```

- When comparing incompatible values, the result is always **false**.

```
…

var Test;

…

Test = 2 and true;

…
```

```
The Variable Test is declared

The following may be placed in an event or function

Test now contains false, because 2 is not a logical value!
```

- When combining Logical values with Text or Number, Logical values become 1 for **true** and 0 for **false**. However, 1 and 0 cannot be converted to Logical values automatically.

```
…

var Test;

…

Test = 5 + true;
Test = 1 and 1;
Test = (2-true==1);

…
```

```
The Variable Test is declared

The following may be placed in an event or function


Test now contains 6, because true was converted to 1!
Test now contains false, because 1 cannot be converted to true!
Test is now true, because true was converted to 1, 2-1 equals 1,
comparing 1 with 1 (1==1) is true.
```

Functions and Return Values

We mentioned functions earlier. Now we will see how to use them.

A function can be used as a single command, or as a command which returns a value. You call a function by writing its name followed by parenthesis which includes values passed as Parameters (also known as Arguments).

```
…

var funcVar;                                Declare a variable, visible for the entire Scenario

…

function Add(val1, val2)                    Declare a function called Add
begin
   funcVar = val1 + val2;                   funcVar now contains the summation of val1 and val2
   return;                                  Exit the function. This is optional!
end

…                                           Inside an Event or a different Function:

Add(20, 15);                                After this command, funcVar has the value 35

…
```

A Function may also have a result. The result is specified with the **return** keyword. When a Function has a return value, calling it can be used as a value.

```
…

var testVar;                                Declare variables for testing purposes
var testVar2;
var testVar3;

…

function Add(val1, val2)                    Declare a function called Add
begin
   var tempResult;                          Declare a temporary variable only usable in this Function
   tempResult = val1 + val2;                Add val1 and val2 together into tempResult
   return tempResult;                       Return the value of tempResult as a result
end

…

testVar = Add(20, 15);                      After this command, testVar has the value 35
testVar2 = Add(2, 50);                      testVar2 now has the value 52
testVar3 = 10 < Add(8, 7);                  testVar3 now has the value true, because 10 is less than 15
…
```

Because a large Scenario can have many Variables, this might become overwhelming. Thus, unless there is a reason, it is preferred to return the result of a function rather than assigning it to a Scenario Variable. Also, if a Variable is only usable in a Function (such as tempResult) or an Event, it is preferable to declare it there so that in the end the Scenario can be understood more easily.

Note: The **return** keyword is optional. You may use **return;** to simply exit a function.

Note: When a Function does not specify a return value, the return value is automatically the special value **null**.

Note: When a Variable in a Function or a State changes, its value is reset and it is not remembered!

If-Statements

In order for a Scenario to become more interactive, its author should use If-Statements. The If-Statements alter the flow of the commands by checking whether a logical prerequisite is true or false.

Consider the following example. A patient suffers from a condition which requires the doctor to give him a special kind of medication (say, RedPill) to survive. However, he may be only given two 5mg doses of RedPill. Providing more than that will simply kill the patient. This is how you would do this in Scribulance:

```
…

var redpill = 0;                                    Count how many redpill doses the patient was given

…                                                   In some State

option "Provide 5mg of Foolazol" do
begin
   if redpill < 2 then                              Check if the patient was given less than 2 doses
   {
      redpill = redpill + 1;                        Increase the counter of doses
   }
   else                                             This happens if redpill is equal or greater than 2
   {
      //Kill the patient                            Whoops!
      //Commands omitted
      //More on this later
   }
end

…
```

Note: Remember that using the curly brackets **{** and **}** are exactly the same way as using **begin** and **end**, and vice-versa!

In the above example, when the user is in the State in which this Option Event belongs to, he has the available option to Provide 5mg of Redpill. When the user clicks on this option everything is fine, but if they click more than twice, the patient dies.

Note: **else** statements are optional, and they must always succeed an **if <condition> then** statement!

Finally, If-Statements may be nested.

```
…

if variable == true then
{
   // commands
   if variable2 == true then
   {
      //commands
   }
```

```
    else
    {
        //commands
    }
}
else
{
    if variable3 == true then
    {
        //commands
    }
}
end

…
```

Conditional Options

Conditional Options are similar to If Statements. They are ordinary Options, except they come with a condition. When that condition is **false**, they are disabled to the user. When an Option is disabled, it appears grayed out and clicking it does not do anything.

Consider the following example: There is a machine with two on/off switches. In order to use it, both switches must be on. This is how you would do this in Scribulance:

```
…

var switch1 = false;                    Whether switch1 is on (true) or off (false)
var switch2 = false;                    Whether switch2 is on (true) or off (false)

…                                       In some State

option "Activate Switch 1" do
begin
   switch1 = not switch1;               Invert the logical value of switch1
end
option "Activate Switch 2" do           This happens if foolazol is equal or greater than 2
begin
   switch2 = not switch2;               Invert the logical value of switch2
end

option "Use Machine"                    Another option
when switch1 and switch2 do             Option condition, check if both switches are on
begin
   //Use the machine
   //Commands omitted
end

…
```

The Use Machine Option is only available if both switches are activated. The above example has a similar effect with the following:

```
…

option "Use Machine" do
begin
if switch1 and switch2 then             Check if both switches are on
   begin
      //Use the machine
      //Commands omitted
   end
end

…
```

The difference is how it appears to the user. In the first case the Option is grayed out until both switches are on. In the second case the Option appears valid to the user, but clicking it still does not do anything.

State Control Commands

State Control Commands are used to change from a current State to another. There are two available commands: The goto and the jumpto command. They achieve the same result, with one important difference. Whenever the author of a Scenario uses the goto command, the Exit Event of the current State and the Enter Event of the target State are executed in that order. On the contrary, using the jumpto command does not execute those Events.

Note: The Enter Event of the first State, if it exists, is always called when the scenario begins!

Note: Even when using the **goto** command, any related Enter and Exit Events are still optional.

Here is an example utilizing the above:

```
scenario "State Controls" startState;          Say that the first state will be startState

var testVar1;                                  Declare testing Variables
var testVar2;

state startState                               Declare the State called startState
begin
   on enter do                                 Enter event for startState
   begin
      testVar1 = "Enter Start";                Set testVar1 to "Enter Start"
   end

   option "goto other" do                      Declare an Option
   begin
      goto otherState;                         Activate the State otherState, execute events
   end

   option "jumpto other" do                    Declare an Option
   begin
      jumpto otherState;                       Activate the State otherState, don't execute events
   end

   on exit do
   begin
      testVar1 = "Exit Start";                 Set testVar1 to "Exit Start"
   end
end

state otherState                               Declare the State called otherState
begin
   on enter do                                 Enter Event for otherState
   begin
      testvar2 = "Enter Other";                Set testVar2 to "Enter Other"
   end

   option "Go back" do                         Declare an option
   begin
      goto start;                              Activate State start, execute events
   end

   on exit do                                  Exit Event for otherState
   begin
      testvar2 = "Exit Other";                 Set testVar2 to "Exit Other"
   end
end
```

When the user starts this Scenario, testvar1 contains "Enter Start" and testvar2 is empty. Assume the following actions are taken in this order:

1. The user clicks "jumpto other". The active State is now otherState. Nothing else happened, because the author used jumpto instead of **goto**.

2. The user now clicks "Go Back". Because the author used **goto**, the Exit Event of otherState is executed, and testVar2 now contains "Exit Other".

3. The user now clicks "goto other". Because the command here is **goto** instead of **jumpto**, testVar1 now contains the value "Exit Start" and testVar2 now contains the value "Enter Other".

4. The user now clicks "Go Back", testVar2 now contains "Exit Other" and testVar1 contains "Enter Start".

Generally, you will only need the **goto** command.

Time Events

There are two kinds of Time Events: State Time Events and Dynamic Time Events.

State Time Events are Events that are executed after a specified time a State is activated.

```
…

when 0:10 do                                    In a State
begin
   //set of commands                            Time event to be executed after 10 seconds
end

when 1:30 do                                    Time event to be executed after  1 minute and 30 seconds
begin
   //set of commands
end

…
```

This is quite straightforward. There is, however, one thing that you should keep in mind: State Time Events are invalid after their State is deactivated. This means three things:

1. If, for example, a State Time Event is set to happen 10 seconds after it is activated, but the State changes before this happens, this Time Event will not happen.

2. Assume a State Time Event is set to happen10 seconds after its State is activated, but the State changes 4 seconds after this happens. Returning to this State will not let the Time Event continue and trigger in 6 seconds. It will be executed in 10 seconds, as its timer is reset to 0.

3. When a Time Event is executed, and then the State exits but it is activated later on, the Time Event will be executed again (provided the State is still active).

None of that applies to Dynamic Time Events. Activating them is a command and they don't belong to a specific State. Also, the same Dynamic Time Event might be pending more than once at the same time. They also require a Function in order to be used.

Dynamic Time Events are implemented like this:

```
…

function eventToBeCalled()                      Declare a Function
begin
   //set of commands
end

…

option "Time Event in 2 seconds" do             Declare an Option
begin
   call eventToBeCalled() when 0:2              Dynamic Time Event in 2 seconds, execute eventToBeCalled()
```

```
end

…
```

The Dynamic Time Event won't happen until the user clicks on the option shown above. Assume the following actions are taken by the user.

1. The user clicks on "Time Event in 2 seconds" at 20 seconds after the Scenario had begun.
2. The user clicks again at 21 seconds after the Scenario had begun.

Then what will happen is that the function eventToBeCalled() will be executed at the 22 second mark (because of step 1) and it will be executed again at the 23 second mark (because of step 2).

Because of that, Dynamic Time Events are more versatile but overusing them might make your Scenario more complicated.

The Functions called by Dynamic Time Events may also have Parameters.

```
var testVar;                                Declare a test Variable

…

function event(left, right)                 Declare a Function
begin
   testVar = left + right;
end

…

option "Time Event in 2 seconds" do         Declare an Option
begin
   call event(2, 2) when 0:2;               Dynamic Time Event in 2 seconds, execute event(2, 2)
end

…
```

There is one more thing you have to keep in mind, however. Consider this example:

```
var testVar;                                Declare a test Variable

…

function event(left, right)                 Declare a Function
begin
   testVar = left + right;
end

…
state testState
begin
   var stateVar = 2;                        Declare a State Variable

   option "Time Event in 2 seconds" do      Declare an Option
   begin
```

```
      call event(stateVar, 2) when 0:2;
   end

   option "Change State Variable" do
   begin
      stateVar = 4;
   end
end

…
```

Dynamic Time Event in 2 seconds, execute event(stateVar, 2)

This is more complicated than the previous example. This time we use a Variable as a Parameter. The value of that Variable might change any time after the Time Event starts counting, but before it is executed. However, the value that will be called will be the one the Variable had when the countdown begun, and not when the function is executed.

Assume the following actions are taken by the user:

1. The user clicks on "Time Event in 2 seconds" at 20 seconds after the Scenario had begun. The Variable stateVar currently holds the value

2. The user clicks immediately after that, before the 21st second, the option on "Change State Variable". Now stateVar currently holds the value 4.

3. The user clicks at the 21st second the option "Time Event in 2 seconds".

What will happen is that at the 22 second mark, testVar will have the value 4 despite the fact that stateVar had changed its value before. At the 23rd second, however, testVar will gain the value 6.

Annotations, Built-in Variables and Functions

In order to create Scenarios that are complete, one needs to be able to display some sort of information to the user. The only way to do this is by using Annotations, Built-In Variables and Functions. They are special in two ways:

1. You cannot declare them, as they are already declared.
2. They are connected with the Game itself; changing / calling them will produce a visible effect.

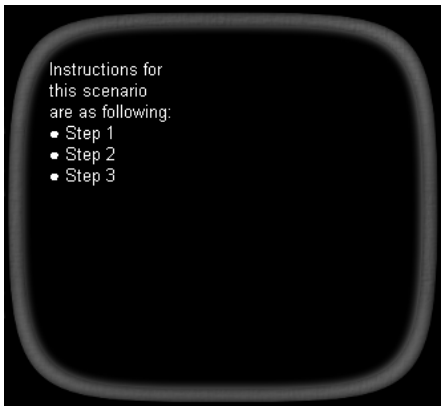
The Annotations can any of the following: History, Instructions and Objectives. Setting a value to them may look like setting a value to a Variable, but Annotations are NOT Variables. You may only set a value to an Annotation immediately after the Scenario Declaration, or immediately another Annotation. You may declare the three Annotations at any order, and declaring an Annotation is optional. The only restraint is that you may declare the same Annotation only once.

Note: You may declare Variables named as one of the three possible annotations, but changing their value will not actually change the annotations themselves.

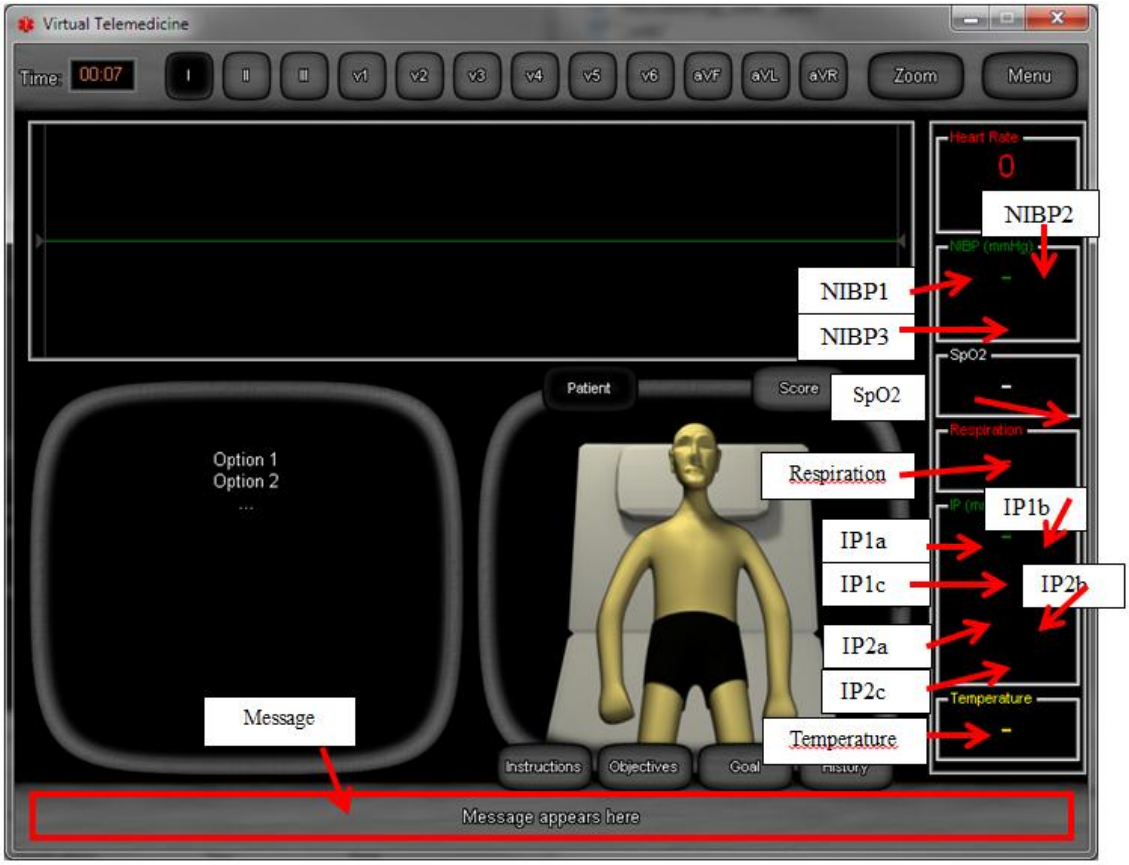
Keep in mind that the text data of an Annotation can span across multiple lines. Using the special character #, an Annotation will appear in the game as if the line breaks. Moreover, the special character ~ will appear like a bullet. Thus, the following declaration:

```
…

Instructions = "Instructions for
this scenario#are as following:
~ Step 1
~ Step 2
~ Step 3";

…
```

Will appear as such:



The Built-In Variables are shown here:



Note: Remember that Variables, as with any other name, are case-insensitive. Thus, message is the same as Message.

Assigning a value to any of the Variables noted in the figure, the corresponding value will appear to that position noted by the arrows.

For example, the following command will display the message "Hello World" on the bottom of the game screen:

```
…
Message = "Hello World";
…
```

```
In a Function or Event
Assign "Hello World" to the built-in Variable Message
```

The Built-In Functions are:

- WinGame(message)
  Win the game and display the message to the user.
- LoseGame(message)
  Lose the game and display the message to the user.

- EndGame(message, minimum_score)
  Finish the game and display the message to the user. The game is won if the score is greater or equal to minimum_score, or otherwise lost.

- GetScore()
  Returns the current value of the score

- SetScore(Score)
  Sets the value of the game score to that of Score. This should typically only be used at the beginning of a scenario and use Grade(…) for other cases instead.

- AddScore(Score)
  Add Score to the game score. This Function should generally be avoided, as it doesn't produce any visible results. The use of Grade(…) is encouraged instead.

- Grade(Score, remark)
  Add Score to the game score. This will appear in the Scoring information, along with remark to explain the reasoning behind the updated score. Score can be negative or 0. If it's positive, it appears in green, if negative in red. If it's zero the remark appears white, but the number 0 is not shown.

- SetColor(color_name)
  Changes the color of the patient. Valid values for color_name are:

- SetAnimation(animation_name)
  Changes the animation of the patient. Valid values for animation_name are:

  "BreatheHard" "BreatheNormal" "BreatheSlow" "ChestPain" "Dead"

  "Default" "Headache" "Stomachache"

- SetColor(color_name)
  Changes the color of the patient. Valid values for color_name are:

  "Normal" "Pale" "Blue"

- SetColorVal(red, green, blue)
  Changes the red / green / blue values of the patient's color. Each Parameter gains a value between 0 and 255. For example, a value of (0, 0, 0) is completely black and a value of (255, 255, 255) is completely white. (255, 0, 0) is red, (0, 255, 0) is green and (0, 0, 255) is blue. Immediate values will produce immediate results.

- SetECG(ecg_name)
  This Function changes the ECG signal displayed by the ECG graph in the game. It also automatically changes the Heart Rate value to correspond to the signal. Valid values for ecg_name are:

"180"	"150"	"120"	"90"	"75"
"60"	"45"	"30"	"A.FIB"	"A.FLUTTER"
"A.PAUSE"	"ARTEFACT"	"AUTO"	"AV-BL.I"	"AV-BL.II"
"BIGEM"	"COUPLET"	"DEM.PACER"	"INTERF"	"NONE"
"PACER"	"PACER.MAL"	"R-ON-T"	"RUN"	"SIN.ARRHY"
"ST.DEPR"	"ST.ELEV"	"SVPB.S"	"V-FIB"	"V-RHYTHM"
"VBP.MULT"	"VBP.UNI"	"VTACH"	"RBBB"	

- SetECGTo(ecg_name, rate)
  This Function changes the ECG signal displayed by the ECG graph in the game like in SetECG. However, it also re-adjusts the signal so that the average heart rate of the ECG matches that of rate.

- SetSignal(sig_name)
  Sets the signal type. It has the same result as clicking the signal type buttons on the top of the Game screen. Valid values for sig_name are:

"I"	"II"	"III"	"V1"	"V2"
"V3"	"V4"	"V5"	V6	AVF
"AVF"	"AVL"	"AVR"		

- Random()
  Returns a random value between 0.0 and 1.0. This can be used for actions that have a chance to succeed.

- RandomInt(start, end)
  Returns a random value between start and end. Contrary to Random(), however, this Function only returns Integer numbers.

- Round(number)
  Returns a rounded version of number.

- Floor(number)
  Returns the floor of the number, which is the largest integer that is less or equal to number.

- Ceiling(number)
  Returns the ceiling of the number, which is the lowest integer that is greater or equal to number.

- Trunc(number)
  Truncates the decimal part of the number and returns it.

- ToText(number)
  Converts and returns a number to Text.

- ToNumber(text)
  Converts and returns a text to a number

Note: Despite the fact that Text values are case-sensitive, Built-In Functions have a special mechanism which ignores capitalization. Therefore, SetColor("Pale") will produce the same result as SetColor("PALE").

Built-in Functions are called like normal Functions. Here is a useful example:

```
…

option "50% chance to live or die" do
begin
   if Random() >= 0.5 then
   begin
      WinGame("Risk and win!");
   end
   else
   begin
      LoseGame("Risk and lose!");
   end
end

…
```

```
In a State


This has a 50% chance to be true

Win the game


Lose the game
```

The above example is an Option which, when clicked, may save or kill the patient with a 50% chance.

Additional Useful Information

There are certain things that you might find useful in writing Scenarios.

1. **Whitespace does not matter**
   Whitespace is a word for spaces, enter, tab, etc. The way you use whitespace doesn't matter as long as you separate keywords with at least one whitespace character and as long as you end each command with a semicolon (;).

   For example, this code is perfectly fine and will work as intended.

```
…                                          In a State

option ".." do begin var test; Message
= "Hello";

test =
2 + 4;        end

…
```

   It is important, however, to maintain a good structure for your code! Case in point: Compare this this example to any of the examples above. Which is more readable and understandable?

2. **Variables cannot be declared anywhere.**
   A Variable can only be declared in any of the following situations:

   - Immediately after the declarations of all Annotations or Scenario declaration (in the situation an Annotation is not declared)
   - At the beginning of a Function
   - At the beginning of a State
   - At the beginning of an Event

3. **With the exception of States, begin-end blocks are sometimes optional**
   States are required to have a begin-end block, even when they are empty. On the other hand we have Events, Functions, If-Statements and Else-Statements that don't require a begin-end block. When a begin-end block is not used, however, it is assumed that there is only ONE command belonging to those statements. Keep in mind that an entire If-Then-Else statement counts as one single command in these cases.

   Take a look in this example:

```
…                                          In a State
```

```
option "1" do
begin
   if true then
      Message = "Hello!";
   else
      Message = "";
end

option "2" do Message = "Hello!";

option "3" do if true then
   Message = "Hello!";
else
   if true then Message = "Hello!";

option "4" do
if true then Message = "Hello!";
else
   Message = "Hello!";
   if true then Message = "Hello!";
…

option "5" do
begin
var test = true;
if test == true then
   Message = "Hello!";
   test = false;
else
begin
   Message = "Hello!";
end

…
```

```
This is valid!

This is valid!

This is valid!

This is NOT valid. This statement belongs nowhere.

The If-Statement ends here!
Not here
This is NOT valid. It does not belong to an If-Statement.
```

With the exception of options "4" and "5", the above code is correct and will work as expected. Of course, the code is not very practical. For example, **if true then** will always be executed and the **else** part will be ignored. It is, however, syntactically correct.

4. **Be wary of assignment instead of comparison**

   When using a single equals sign (=) it is an assignment. When using a double equals sign (==) it is a comparison. Do not use single equal signs in If-Statements, because it will produce unexpected results.

   Consider this example:

```
…

option "..." do
begin
   var test = false;
   if test = true then
      Message = "True";
   else
      Message = "False";
end

…
```

```
In a State
```

You would expect that clicking on "..." would always show "False" to the Game screen. However, that won't happen! It will always print "True" instead!

Why? Because when we used a single equals sign (=) in the condition of the If-Statement, the value **true** was assigned to the Variable test, and after that the value of test was checked by the If-Statement (which is, unquestionably, **true**).

Extending Scribulance

This is an advanced topic and it is not required to write scenarios for Virtual Telemedicine.

Scribulance may be adapted for use by other applications, by using it as a .NET library. You may download the binaries at:
http://bytefreaks.net/wp-content/plugins/download-monitor/download.php?id=26
This is very straightforward and customizable. It is done by defining the three built-in aspects tailored for your application: Built-in Functions, Built-in Variables and Built-in Annotations. There are three steps required:

1. **Make the compiler recognize your built-in components**
   A text file called extern.dat is used to define these and must be placed in the same directory as the binaries.

2. **Implement built-in functions in your .NET application**
   If a function exists in extern.dat but is not implemented, a script using that function will compile, but running it will crush your application when that function is called.

3. **Run and update scenarios through your application**
   This task is also very simple, because the runner executes all code automatically.

The file extern.dat is divided in three sections and could be defined as such:

```
var Message; var Temperature;
var AnotherBuiltInVariable;
```

```
SetECG$1~
SetECGto$2~
AddThreeNumbers$3~
```

```
$instructions
$history
$objectives
$anotherAnnotation
```

At first, variables are defined exactly like in an ordinary scenario. After that, functions are defined by their name, followed by the symbol $, followed by the number of arguments, followed by the symbol ~. And last, annotations are defined by the symbol $ followed by the annotation name.

Implementing functions is almost just as easy as implementing any method in your .NET application, with a bit more verbose syntax. However, it is just as powerful, and you can even reference application-specific components through your built-in functions.

Assuming you reference the library in your code:

```
using Runbulance;
```

You can then implement a function by implementing an ordinary method in your application, as long as it has the following signature:

```
Value MethodName(IVariable[] args) {
	return Value.Null;
}
```

You can then register a function like so:

```
FunctionRegistry.Register("FunctionName", 1, MethodName);
```

Where "FunctionName" is the name of the function as defined in extern.dat and can be used in a scenario, and the number 1 is the number of arguments.

In another example which uses arguments, the function AddThreeNumbers can be registered like so:

```
//Implementation
Value MethodNameIrrelevant(IVariable[] args) {
	int? A = args[0].Evaluate().ToReal();
	int? B = args[1].Evaluate().ToReal();
	int? C = args[2].Evaluate().ToReal();
	if (A.HasValue && B.HasValue && C.HasValue) {
		return new Value(A.Value + B.Value + C.Value);
	}
	else {
```

```
        }
}

//...
//Registration
FunctionRegistry.Register("AddThreeNumbers", 3, MethodNameIrrelevant)
```

The Value class represents a wrapper of any value acceptable by Scribulance. You can create a Value by using int, float, bool or string as a constructor parameter. Moreover, you may also use the read-only values Value.Null, Value.Illegal, Value.True and Value.False.

Having customized Scribulance, you can now use it in your application by creating an instance of the Runbulance.Runner class:

```
Runner runner = new Runner("scenario_filename");
```

Note: This assumes that the scenario is already compiled!

You can then access the description of the scenario with:

```
string desc = runner.Name;
```

Or access an annotation:

```
string info = runner.Info("AnnotationName");
```

Or get/set the value of a variable, using the index operator:

```
runner["VariableName"] = new Value(10);
Value val = runner["VariableName"];
```

Note: Built-in variables are guaranteed to always work, but scenario-specific variables are not!

Or get a list of the options:

```
List<Runner.Option> list = runner.QueryOptions();
```

Note: Option is a struct which has the members Available and Description.

And then execute an option by its index

```
runner.ExecuteOption(int index);
```

Last, but not least, the Update method is necessary for all time-based events to work:

```
runner.Update(int seconds);
```

Note: Typically, Update should only be called at one second intervals with a parameter of 1.

Finally, an independent application would typically need to be bundled with the compiler. One needs three executables to compile: Prebulance.exe, Globulance.exe and Synbulance.exe in that order. The scenario source filename must be the only argument to Prebulance.exe and pipe the output to Globulance.exe and finally Synbulance.exe. The final standard output is the intermediate language which can be executed by the runner and the error output is any errors that might have been encountered.

For reasons of convenience, there is a fourth executable supplied which does exactly that. It needs only to be provided with the filename as an argument and it outputs two files: The compiled result (filename: output) and any errors (filename: error) that might have been encountered. Whenever the error file is empty, the scenario compiled successfully.

Note: Output is always created, even in the case of an error. When an error occurs the output must be discarded.

Παράρτημα Β

Virtual Telemedicine: A Serious Game to Develop Problem Solving Skills in Medical Education

Authors:
A. Antoniades, C. Marangos, K. Efthyvoulos, I. Nicolaidou, E. Dafli, P. Bamidis, and C. Pattichis

Publication: EDULEARN12 Proceedings
Pages: 3612–3619
Year: 2012

VIRTUAL TELEMEDICINE: A SERIOUS GAME TO DEVELOP PROBLEM SOLVING SKILLS IN MEDICAL EDUCATION

Athos Antoniades[1], Charis Marangos[1], Efthyvoulos Kyriacou[2], Iolie Nicolaidou[1], Eleni Dafli[3], Panagiotis Bamidis[3], & Constantinos Pattichis[1]

1. [1]*Department of Computer Science, University of Cyprus (CYPRUS)*
2. [2] *Department of Computer Science & Engineering, Frederick University (CYPRUS)*
3. [3] Lab of Medical Informatics, Aristotle University of Thessaloniki *(GREECE)*
4.

athos.antoniades@stremble.com, charismarangos@gmail.com, e.kyriacou@frederick.ac.cy,

iolie.nicolaidoue@stremble.com, elendaf@yahoo.com

Bamidis@med.auth.gr, pattichi@cs.ucy.ac.cy

Abstract

Serious games have the potential to be an important teaching tool for both formal and informal education because of their affordances of interactivity and motivation, engaging users, providing a platform for active learning, being customized to learners, providing immediate feedback and including immersive activities. Serious games for medical education is a growing domain. Across the healthcare sector, there is growing interest in improving and sustaining interaction and engagement using game technologies. Game environments provide a safe and controlled setting within which players can learn in an engaging way. Health games for practitioners, such as doctors and nurses, tend to be simulation-based and used for training.

This paper is a report on the design and development of a serious game called “Virtual Telemedicine”. The game responds to the need to train doctors for problem-solving in real-life situations of medical care through a telemedicine system. In the prototype scenario, the user, a practicing doctor, responds to a medical emergency situation to treat a virtual patient who is located remotely through the use of a telemedicine system by taking specific actions and examining their results based on immediate feedback provided by both the system and the virtual patient. Virtual Telemedicine was created by repurposing educational content as part of the meducator Best Practice Network. mEducator enables specialized state-of-the-art medical educational content to be discovered, retrieved, shared and re-used across European higher academic institutions.

The game makes use of data from an electrocardiogram (ECG), a transthoracic (across the thorax or chest) interpretation of the electrical activity of the heart over a period of time, as detected by electrodes attached to the outer surface of the skin and recorded by a device external to the body. Furthermore, the patient’s blood pressure and oxygen saturation levels are reported along with video of the patient and written communication. The game also supports the request of specific diagnostic tests a virtual nurse could be applying to the patient locally and reporting the results through the telemedicine system in written form.

The virtual telemedicine game implements a scripting language that enables the creation of custom scenarios for the game. These scenarios are currently text based using a user friendly high level scripting language developed for the purposes of this game. The game was created using the programming language C# with XNA for the game, C with Flex/Bison for the compiler and C# for the Runner, that runs the scripts. The three pieces are separate and the user runs the

first piece. For testing and evaluation purposes, several scripts that incorporated the features of the scripting language were implemented. The educational game “Virtual Telemedicine” is currently working as a standalone application on PCs with the Windows operating system. The game can be shared and repurposed (through changing its scenarios) through several instantiations of mEducator.

As part of future research and development work focusing on the educational evaluation of this serious game, usability testing sessions with practicing doctors will be scheduled to solicit the learners’ feedback and input on the design of the game. Suggested changes will be incorporated in an updated version of the game. A direction for further research is to examine whether learning will be transferred beyond the game context and how, and what is the retention rate of the knowledge acquired by medical students who will use the game.

Keywords: serious games, medical education, e-learning, virtual patients, problem-solving.

2 INTRODUCTION

7.4 The importance of serious games in medical education

Serious games have the potential to be an important teaching tool for both formal and informal education because of their affordances of interactivity and motivation, engaging the user, providing a platform for active learning, being customized to the learner, providing immediate feedback and including immersive activities. According to [1], serious games are defined as “games with an educational intent… that are engaging…have an underlying pedagogy and where learning can be implicit or explicit” (p.5). The learning outcome is dependent upon an appropriate pedagogy and the underlying game mechanics and how the content is integrated into the game so the learning is intrinsic to play [1]. Games are considered as increasingly effective learning tools, particularly when embedded effectively into practice [2].

Developed using computer game technologies more often associated with entertainment, serious games provide a platform for combining high-fidelity graphics and sound with engaging content, novel interfaces, and serious purposes. Meta-reviews on games and simulations have begun to synthesize findings from the literature identifying games as particularly motivating and engaging for learners of all ages [2]. A recent study focused on medical student respondents to investigate their experiences and attitudes and examine whether they warrant the development of new media teaching methods in medicine. The results of the study showed that medical student respondents (n=271) including many who did not play video games, held highly favorable views about the use of video games and related new media technology in medical education [3].

Research showed that among the advantages of serious games is a higher level of retention of material (Magennis & Farrell, 2005, as cited in [1]). Serious games allow learners to experience situations that are impossible in the real world for reasons of safety, cost or time [4]. Games can also support the development of a number of different skills, such as analytical and spatial skills, strategic skills, learning and recollection capabilities, psychomotor skills, and visual selective attention [5].

Utilization of serious games for medical education is a growing domain. Health games for practitioners, such as doctors and nurses for example, tend to be simulation-based and used for training [1]. [2] refers to the example of surgical training as one application of simulations in the medical domain to support training and learning. Another application of simulations in medical education refers to the use of online avatar-based training simulations to motivate people who exhibit signs of post-deployment stress to seek help [6]. Moreover, advanced medical simulators have been used to facilitate surgical and endoscopic training and thereby improve patient safety [7].

Current trends in serious game research and development for healthcare focus on professional training, improving therapeutic outcomes for patients, and promoting awareness of health-related issues to a broader public [8]. Game environments provide a safe and controlled setting within which players can learn in an engaging way, whether they are exploring new clinical

techniques, undergoing rehabilitative activities, or being exposed to health-related issues. An example of a serious game implementing an immersive virtual learning space, which was developed for training health care professionals in clinical skills is Pulse!! - The Virtual Clinical Learning Lab. In this game graphics recreate a lifelike, interactive, virtual training environment in which civilian and military heath care professionals practice clinical skills in order to better respond to injuries sustained during catastrophic incidents, such as combat or bioterrorism. The game is designed to support a range of the training needs nurses and medical professionals require [2].

Another example refers to the use of Second Life, a virtual three-dimensional immersive learning environment in medical education for the administration of mock oral examinations to emergency medicine residents. In research involving this particular learning environment, the examinee managed the case while acting as the physician avatar and communicated via headset and microphone from a remote computer with a faculty examiner who acted as the patient avatar. The results of the evaluation of this simulation-based environment by twenty seven residents who participated in the virtual oral examination provided evidence that the application of Second Life virtual simulation technology can be a potential alternative to traditional mock oral examinations for emergency medicine residents [9].

Positive impacts have also been reported by [7], who implemented advanced medical simulators to facilitate surgical and endoscopic training and found a positive correlation between experience in computer games and performance in endoscopic simulation by medical students. The better performance of gamers was attributed to their three-dimensional perception experience from computer gaming.

7.5 The meducator Best Practice Network

There is evidence to suggest that doctors who train medical students should consider incorporating games and simulations to their practice [9]. For this purpose, a serious game for medical students called “Virtual Telemedicine” was developed by repurposing existing educational content (power point presentations, simulation software, virtual patients scenarios) in the context of the meducator Best Practice Network to teach students how to respond to medical emergency situations through the use of a telemedicine platform. The game responds to the need to train medical students for problem-solving in real-life situations of medical care through a telemedicine system. In the prototype scenario, the user, a practicing doctor, responds to a medical emergency situation to treat a virtual patient who is located remotely through the use of a telemedicine system [10, 11] by taking specific actions and examining their results based on immediate feedback provided by both the system and the virtual patient. Virtual Telemedicine was created by repurposing educational content [12] as part of the meducator Best Practice Network.

The mEducator Best Practice Network (BPN) is an EU-funded project, which implemented and critically evaluated existing standards and reference models in the field of e-learning to enable specialized state-of-the-art medical educational content to be discovered, retrieved, shared and re-used across European institutions [13, 14]. mEducator includes both traditional and user-generated content and addresses several learning contexts ranging from traditional instructional teaching to active learning and experiential teaching/studying approaches. It furthermore includes many different content types, ranging from text to exam sheets, algorithms, teaching files, computer programs (simulators, serious games) and interactive objects (like virtual patients and digital tracings of anatomies), while it covers a variety of tools.

Two contemporary ways of achieving content sharing have been developed: mEducator2.0, a solution based on Web2.0 technologies and mEducator3.0, a solution based on Semantic Web Services. Both mEducator solutions have been tested by the target user groups of the project, i.e., students, educators, doctors, and health professionals [15]. To provide more context for the sharing and repurposing of the Virtual Telemedicine game additional information on mEducator 3.0 is given.

The mEducator 3.0 solution is designed around a federated architecture based on a service-oriented application framework and the use of semantic technologies. The utilized framework is fundamentally based on the Semantic Web Services-oriented e-learning architecture [16, 17] and the emerging Linked Data and Linked Services paradigms [18]. This solution fundamentally exploits semantic representations of data and services to provide interoperability between e-learning repositories spread across the Web. This stems from descriptions of the content with metadata, which also follow linked data principles, as well as classical vocabularies of metadata [19, 20]. There are four instantiations of the mEducator 3.0 solution, which differ on the type of content management technologies (CMT) they are integrated with; namely, one utilizes Moodle [21, 22], one the Drupal CMS, one the Open Labyrinth platform for Virtual patients and one, Metamorphosis+, is based on the social network Elgg. There is also Melina+, an extended version of Drupal 7, which is offered as an installation profile and enables web site administrators to install a learning management system, which is focused in medical education.

To give an illustrative example of the capabilities that these instantiations provide to users, the mEducator 3.0 Drupal instantiation is described in more detail. As can be seen in Fig. 1, users can explore a large collection of educational objects, create their own educational object including a full metadata description or collaborate with other members of the community (Fig. 1). The Virtual Telemedicine game can be shared and repurposed (through changing its scenarios) through the several instantiations of mEducator.

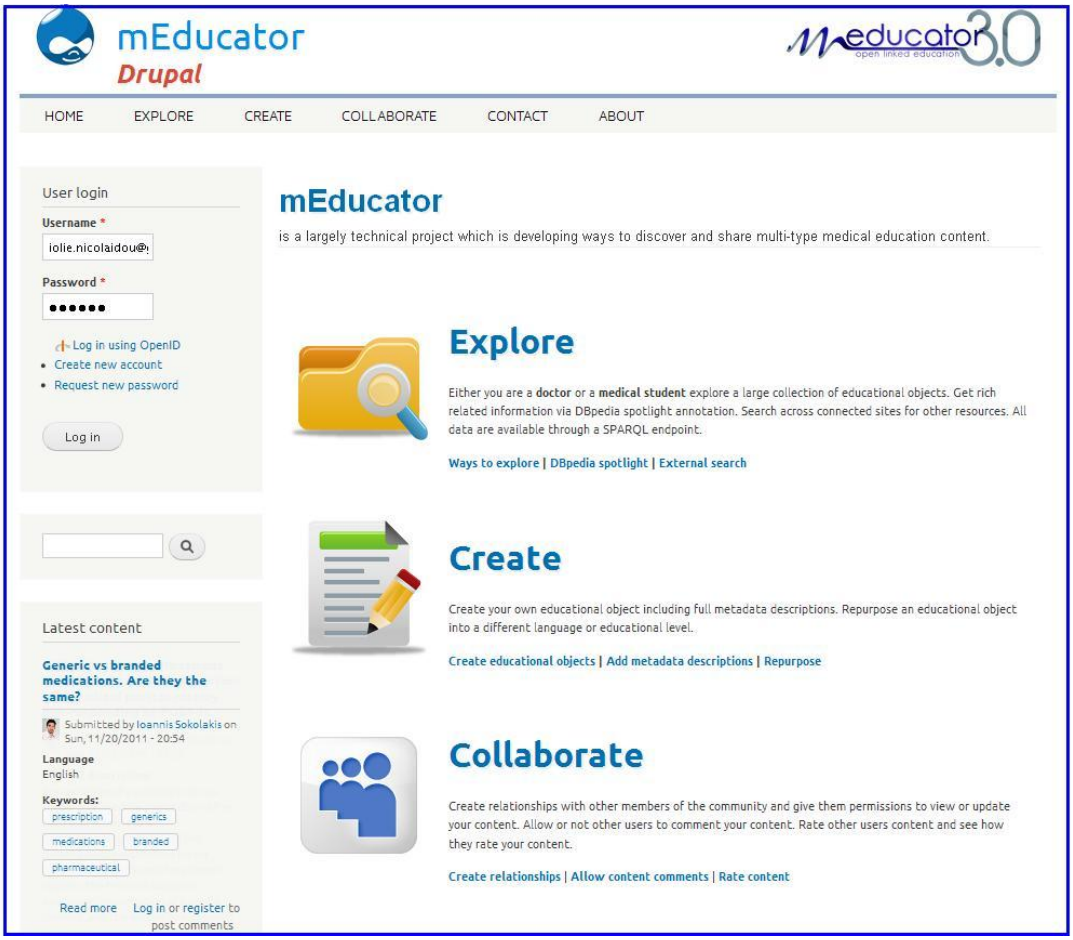


Fig. 1: The mEducator 3.0 Drupal interface, from where users can access and repurpose the Virtual Telemedicine game (through changing its scenarios)

3 THE DESIGN AND DEVELOPMENT OF THE VIRTUAL TELEMEDICINE GAME

7.6 Design of Virtual Telemedicine game

Virtual Telemedicine was designed to support medical students (studying to become doctors, nurses or paramedics) in developing problem-solving skills in different scenarios that can be built into the game. The learning goal of the game is for students to learn how to respond to a medical emergency situation to treat the symptoms of a virtual patient by taking specific actions and examining their results based on immediate feedback provided by both the system and the virtual patient. The Virtual Telemedicine serious game is simulation-based in the sense that it implements "rigorously structured scenarios with a highly refined set of rules, challenges, and strategies which are carefully designed to develop specific competencies that can be directly transferred into the real world" ([1] p.17). Simulations are considered as "facilitators of virtual experiences" ([2], p.11) and are associated with exploratory learning that in this case helps learners rehearse skills for the real-world.

The added pedagogical value of the game relies on the fact that medical students can practice and replay the game in a safe and controlled learning environment until they master the skills without any risk to a real patient. Moreover, the capability is provided to any person with a medical background to add scenarios to the game, whose level of complexity can vary. In this way, increased learner control is provided for motivation purposes as the user can choose among different difficulty levels to determine how challenging the game will be.

The objectives of the design of the Virtual Telemedicine game were to:

a) ensure a high level of fidelity, defined as close resemblance to actual events with the goal of enabling transference

b) incorporate the ability to modify the scenario by doctors and/or add scenarios

c) have a low learning curve, in order for players to learn the game effortlessly

d) allow players to improve through repeated play

e) provide appropriate feedback

f) engage and motivate the learners

g) have a high retention rate

The game makes use of data from an electrocardiogram (ECG), a transthoracic (across the thorax or chest) interpretation of the electrical activity of the heart over a period of time, as detected by electrodes attached to the outer surface of the skin and recorded by a device external to the body. Furthermore, the patient's blood pressure, and oxygen saturation levels are reported along with video of the patient and written communication. The game also supports the request of specific diagnostic tests a virtual nurse could be applying to the patient locally and reporting the results through the telemedicine system in written form.

In the prototype scenario, the user, a practicing doctor, responds to a medical emergency situation to treat the symptoms of a virtual patient by taking specific actions and examining their results based on immediate feedback provided by both the system and the virtual patient. More specifically, the prototype scenario of the game is called "Precordial Pain". As can be seen in Fig. 2, the user is informed that the patient is facing a problem. To take action and save the patient the user can choose one of seven possible options: provide aspirine to the patient, or provide alternative drugs such as Atropine, Morphine, etc. In each case the system provides the user with immediate feedback.

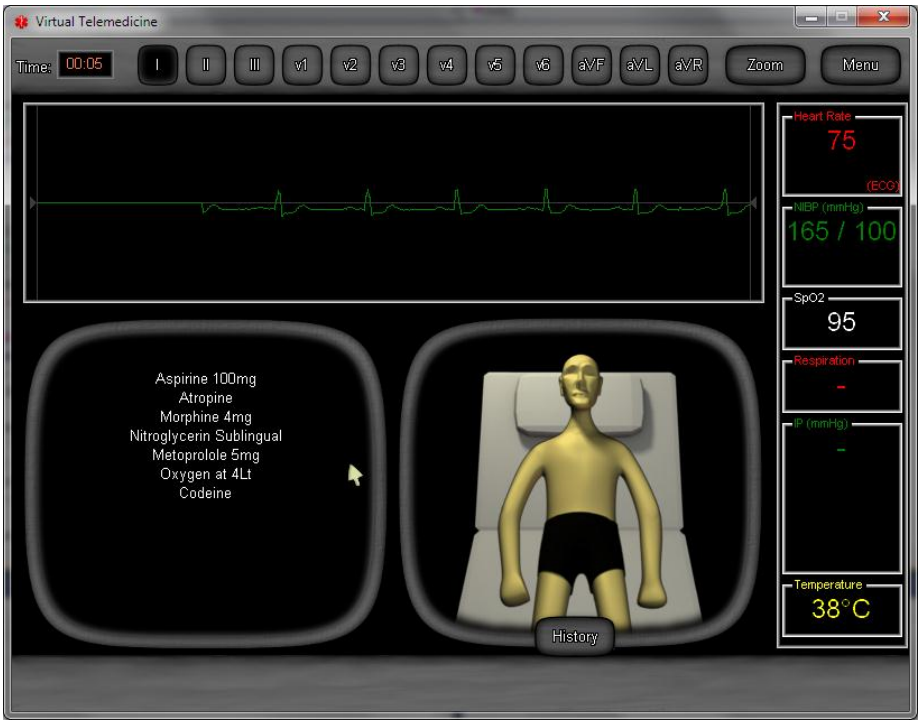


Fig. 2: Screencapture showing the graphical user interface of the Virtual Telemedicine game

Upon successful completion of the game, medical students learn that in this specific scenario where a patient suffers from precordial pain, sensible choices are Aspirine, Morphine, Nitroglycerin Sublingual, Metoprolol and Oxygen. Making the wrong choice may bring negative side-effects. For example, providing Codeine will cause the patient to have increased heart rate and chest pain. After giving the wrong medication, the patient might lose his conscience. In this case, the user is provided with different options that correctly correspond to this new situation. Namely, the user is asked to give a defibrillation treatment. Failure to do so will lead to the death of the patient. The scenario is won after the patient stabilizes and the user gives him the correct medication, which is Clopidogrel.

7.7 Development of Virtual Telemedicine game

The virtual telemedicine game implements a scripting language that enables the creation of custom scenarios for the game. These scenarios are currently text based using a user friendly high level scripting language developed for the purposes of this game. The language is called Scribulance and it is a blend of C and Pascal, but no prior knowledge of programming is required to successfully write a valid scenario.

The game was created using the programming language C# with XNA for the game. In order to support a high level scripting language for developing the game's scenarios a scenario compiler was also created using C with Flex/Bison for the compiler and C# to import the scenarios into the game. For testing and evaluation purposes, several scripts that used the features of the scripting language were implemented to examine whether they worked properly and a detailed command-by-command debugging was conducted to examine that the flow of the code worked as intended.

The educational game "Virtual Telemedicine" is currently working as a standalone application on PCs with the Windows operating system. The game can be shared and repurposed (through changing its scenarios, or creating new ones and compiling them with the included Virtual Telemedicine compiler). It is discoverable through different instantiations of mEducator, such as

mEducator 2.0 and mEducator 3.0 (an instantiation that utilizes Moodle [21, 22], an instantiation that utilizes the Drupal CMS, an instantiation that utilizes the Open Labyrinth platform for Virtual patients, Metamorphosis+, an instantiation that is based on Elgg and Melina+, an extended version of Drupal 7, which is offered as an installation profile and enables web site administrators to install a learning management system, which is focused in medical education.

Research suggests that serious games "are being regarded as interactive technologies that can be used interchangeably with other ICT tools and devices, e.g. social software, to support many different activities and for supporting small and large communities of practitioners and learners" ([2], p.8). In accordance with this suggestion a collaborative component was designed to provide opportunities for reflection after the game, through user interaction. The "Virtual Telemedicine" game will have its own social networking page (via Facebook), designed to invite users to interact with and provide feedback to each other and to collaborate for problem solving, e.g. to discuss different solutions of the scenarios of the game. In this accompanying site, a link to the game as well as installing instructions will be provided to interested users.

4 IN PROGRESS WORK: IMPLEMENTATION AND EVALUATION OF THE VIRTUAL TELEMEDICINE GAME

The evaluation of the Virtual Telemedicine game is part of the work in-progress of the University of Cyprus Medical Informatics Lab. One of the major concerns with respect to using serious games in education is the difficulty in evaluating their effectiveness at achieving their learning goals. To explicitly ensure that learning goals have been reached testing is needed to ensure that players have learned what is intended. Testing can either be internal to the game or external through mediation from teachers who lead reflection based on the users' experience. If testing is internal to the game, there are explicit scoring mechanisms that can be used to assess a game's effectiveness, such as the number of correct answers or the time needed to complete a scenario. In simulations, assessment can also be a comparison of the outcome and the decisions made to reach the solution compared against the ideal, for example an expert's solution [1]. The possibility to build in testing mechanisms internal to the game is being explored with some of the functionality already available.

Usability testing sessions with practicing doctors will also be scheduled to solicit the learners' feedback and input on the design of the game. Their suggested changes will be incorporated in an updated version of the game, which is expected to be used in an implementation scenario involving medical students. The use of serious games needs to be embedded in practice effectively and in accordance with sound pedagogic principles and design [2]. The implementation of the Virtual Telemedicine game in a formal setting will be one of the next steps for this project to evaluate its effectiveness in supporting medical students in learning skills for the "real world".

5 FUTURE WORK

The serious game movement is a trend towards designing and analyzing the use of games and simulations for supporting formal educational and training objectives and outcomes. The movement aims to meet the significant challenge of bringing together game designers and educators to ensure that games that are designed and developed are fun and motivational but also have an educational value [2]. With regard to the development of the mEducator serious game and more specifically its expansion, e.g. the addition of scenarios for problem solving in medical education, there is currently a need for the collaboration among the members of an interdisciplinary group comprised of at least three members: a doctor, who provides the expertise in con-

tent, an instructional designer, who provides the expertise in game design based on theory and pedagogical principles and a programmer, who provides the technical expertise needed for the implementation of the pedagogical requirements of the game. Future work refers to the use of automated ways and the use of simple, structured tree-diagrams for game creators to be able to create their own scenarios via a graphical user interface without the need to consult with programmers for the implementation of their ideas.

A direction for further research is to examine whether learning will be transferred beyond the game context and how, and what is the retention rate of the game. This can be achieved by following up on the practicing doctors who participate in the study to evaluate the Virtual Telemedicine game to examine whether the skills they acquired as part of playing the game have actually been proven useful in real-life situations of patient treatment.

Research shows that serious games can automatically adapt to the player's ability, collect data about their choices over time, provide feedback on these inputs, and provide relevant information as appropriate [1]. Assessment methods may be built into the Virtual Telemedicine game, as part of future work in this area. In this case progress can be assessed through mechanisms such as winning and losing a scenario or completing specific levels.

Novel approaches to game development can potentially enhance learning outcomes and user engagement. However, much research is needed, based on standards for providing valid research evidence in the "games for health" domain to drive the current enthusiasm and activity in this field [23] and to provide the methodologies, frameworks, and development approaches necessary to ensure that this can happen.

REFERENCES

[1] Ulicsak, M. & Wright, M. (2010). Games in education: Serious Games. Bristol, Futurelab. Retrieved from: http://media.futurelab.org.uk/resources/documents/lit_reviews/Serious-Games_Review.pdf

[2] De Freitas,S.(2006). Learning in immersive worlds. A review of game-based learning. Retrieved from: http://www.jisc.ac.uk/media/documents/programmes/elearninginnovation/gamingreport_v3.pdf

[3] Kron, F., Gjerde,C., Sen, A. & Fetters, M. (2010). Medical student attitudes toward video games and related new media technologies in medical education. BMC Medical Education, 10 (50), 1-11. doi:10.1186/1472-6920-10-50

[4] Squire, K. & Jenkins, H. (2003). Harnessing the power of games in education. Insight, 3(1), 5-

33.

[5] Mitchell, A. & Savill-Smith, C. (2004). The use of computer and video games for learning: A *review of the literature*. Learning and Skills Development Agency. Retrieved from: http://www.m-learning.org/docs/The%20use%20of%20computer%20and%20video%20games%20for%20learning.pdf

[6] Albright, G, Goldman, R., Shockley,K., McDevitt, F. & Akabas, S. (2012). Using an Avatar-Based Simulation to Train Families to Motivate Veterans with Post-Deployment Stress to Seek Help at the VA. Games for Health Journal, 1(1), 21-28. doi:10.1089/g4h.2011.0003.

[7] Enochsson, L., Isaksson, B., Tour, R., Kjellin, A., Hedman, L., Wredmark, T. & Tsai-Fellander, L. (2004). Visuospatial skills and computer game experience influence the performance of virtual endoscopy. Journal of Gastrointestinal Surgery, 8(7), 874–880.

[8] Arnab, S., Dunwell, I. & Debatista, K. (2012). Serious games for healthcare: Applications and Implications. IGI GLOBAL.

[9] Schwaab, J, Kman, N., Nagel, R., Bahner, D., Martin, D., Khandelwal, S.,…Nelson, R. (2011). Using Second Life Virtual Simulation Environment for Mock Oral Emergency Medicine Examination. Society for Academic Emergency Medicine, 18(5), 559-562.

[10] Kyriacou, E.C., Pavlopoulos, S. , Berler,A., Neophytou, M.S., Bourka, A. Georgoulas, A., Anagnostaki,A., Karayiannis, D. Schizas, C.N., Pattichis, C.S., Andreou, A.S. & Koutsouris, D.D. (2003). "Multi-purpose HealthCare Telemedicine Systems with Mobile Communication Link Support", *BioMedical Engineering OnLine,* 2 (7).

[11] Pattichis, C.S., Kyriacou, E.C., Voskarides, S.Ch., Pattichis, M.S. Istepanian, R.S.H. & Schizas, C.N. (2002).Wireless Telemedicine Systems: An Overview, *IEEE Antennas* and *Propagation Magazine,* 44 (2),143-153.

[12] Mougiakakou, S., Kyriacou, E., Perakis, K., Papadopoulos, H., Androulidakis, A., Konnis, G., Tranfaglia, R., ecchia, L., Bracale, U., Pattichis, C., & Koutsouris, D., (2011). Towards the Integrated Provision of eHealth Tools and Services: INTERMED and the Experience in the Region of South-East Mediterranean, BioMedical Engineering OnLine, 10 (49).

[13] Nicolaidou, I., Bamidis, P., Antoniades, A., Nikolaidou, M., Pattichis, C. & Giordano, D. (2011). mEducator: Multi-type content sharing and repurposing in medical education. In Proceedings of World Conference on Educational Multimedia, Hypermedia and Telecommunications 2011 (pp. 3740-3745). Chesapeake, VA: AACE.

[14] Bamidis, P., Kaldoudi, E., & Pattichis, C. (2009). mEducator: A Best Practice Network for Repurposing and Sharing Medical Educational Multi-type Content. Leveraging Knowledge for Innovation in Collaborative Networks. IFIP Advances in Information and Communication Technology, 307, 769-776.

[15] Nicolaidou, I., Antoniades, A., Myllari, J., Giordano, D., Dafli, E., Schizas, Ch., Pattichis, C., Nikolaidou, M., Spachos, D. & Bamidis, P. (2011, March). Scenario-based assessment for user-testing of medical educational content-sharing solutions. 6th International Technology, Education and Development Conference**,** INTED2012, Valencia, Spain.

[16] Dietze, S., Gugliotta, A., & Domingue, J., (2008). Supporting Interoperability and Context-Awareness in E-Learning through Situation-driven Learning Processes. Special Issue on Web-based Learning of *International Journal of Distance Education Technologies* (JDET), 2008.

[17] Bratsas, C., Dimou, A., Alexiadis, G., Chrysou, D.E., Kavargyris, K., Parapontis, I., Bamidis, P., & Antoniou, I. (2011). Educational Semantic Wikis in the Linked Data Age: the case of MSc Web Science Program at Aristotle University of Thessaloniki, in Proc. of 1st Int. Workshop on
eLearning Approaches for the Linked Data Age (Linked Learning 2011, in ESWC2011), 2011.

[18] Pedrinaci, C. & Domingue, J. (2010). Toward the Next Wave of Services: Linked Services for the Web of Data, *Journal of Universal Computer Science,* 16 (13), 1694-1719.

[19] Mitsopoulou, E., Taibi, D., Giordano, D., Dietze, S., Yu, H.Q., Bamidis, P., Bratsas,Ch., & Woodham, L. (2011). Connecting medical educational resources to the Linked Data cloud: the
mEducator RDF Schema, store & API; in Proc. of 1st Int. Workshop on eLearning Approaches for the Linked Data Age (Linked Learning 2011, in ESWC2011).

[20] Giordano,D., Kavasidis, I., Spampinato, C.,& Bamidis, P. (2011). Developing controlled vocabularies for educational resources sharing: a case study, in Proc. of 1st Int. Workshop on eLearning Approaches for the Linked Data Age (Linked Learning 2011, in ESWC2011).

[21] Bamidis, P., Konstantinidis, S., Bratsas, C., & Iyengar, S. (2011). Federating Learning Management Systems for Medical Education: a persuasive technologies perspective, In Proc. of Computer Based Medical Systems (CBMS) 2011, IEEE 2011.

[22] Konstantinidis, S., Lakka, C., Bratsas, C., Pappas, C. & Bamidis, P. (2011). Semantic description of digital educational objects in an e-learning environment, In Proc. Of the 2nd

Panhellenic Conference on the introduction and use of ICT in the educational process, pp. 413-422.

[23] Kato, P. (2012). Evaluating Efficacy and Validating Games for Health. Games for Health Journal: Research, Development, and Clinical Applications, 1(1), 1-3.

Παράρτημα Γ

The Instructional Design of Virtual Telemedicine: A Simulation-Based Serious Game in Health-Care

Authors:

I. Nicolaidou, A. Antoniades, C. Marangos, E. Dafli, K. Efthyvoulos, P. Bamidis, and C. Pattichis.

Publication: EDULEARN12 Proceedings

Pages: 5746-5755

Year: 2012

THE INSTRUCTIONAL DESIGN OF VIRTUAL TELEMEDICINE: A SIMULATION-BASED SERIOUS GAME IN HEALTH-CARE

Iolie Nicolaidou[1], Athos Antoniades[2], Charis Marangos[2], Eleni Dafli[3]

Efthyvoulos Kyriacou[4], Panagiotis Bamidis[3], & Constantinos Pattichis[2]

5. *[1]Department of Communication and Internet Studies, Cyprus University of Technology (CYPRUS)*
6. *[2]Department of Computer Science, University of Cyprus (CYPRUS)*
7. *[3]Lab of Medical Informatics, Aristotle University of Thessaloniki (GREECE)*
8. *[4]Department of Computer Science & Engineering, Frederick University (CYPRUS)*

iolie.nicolaidou@cut.ac.cy, athos.antoniades@stremble.com , charismarangos@gmail.com , elendaf@med.auth.gr, e.kyriacou@frederick.ac.cy , bamidis@med.auth.gr, pattichi@cs.ucy.ac.cy

Abstract

This paper is a work-in-progress report on the instructional design of Virtual Telemedicine, a simulation-based serious game in health-care. The problem that this game addresses is the need to train practicing doctors and medical students for problem-solving in real-life clinical scenarios through a telemedicine system. In medical emergency situations a doctor is asked to make important decisions for the treatment of a patient within minutes or even seconds. His actions may result in saving or losing a patient's life. The importance of the development of this game lies on the opportunities it provides to students for practicing their patient-treatment skills in a safe and controlled e-learning environment that simulates real-life conditions but avoids the risks associated with dealing with real patients. The game makes use of data from an electrocardiogram (ECG) and its overall learning goal is for practicing doctors or medical students to learn how to respond to a medical emergency situation to treat the symptoms of a virtual patient by taking specific actions and examining their results based on immediate feedback provided by both the system and the virtual patient. Three scenarios are currently developed in the game: a) Precordial pain, in which the patient faces an acute coronary syndrome, b) Palpitations, in which the patient experiences cardiac arrhythmias and c) Syncope, in which the patient is diagnosed with an atrioventricular block. The paper reports on the instructional design objectives of the game, describes the prototype scenario and outlines the changes made as part of the game's redesign and development to increase its pedagogical value. Some of the changes that were made to the beta-version of the game included the addition of the overall goal of the game, the objectives and context for each scenario, the patient's history, as well as specific instructions to scaffold the user in making a choice among several treatment options. Moreover, instant feedback is now provided to the user as he receives a clearly visible positive or negative score for every one of his actions. In addition, the user can track his score while playing the game. Finally, a detailed report is provided at the end of the game outlining the user's choices, clearly indicating which of them were successful and including the total amount of time he required to solve the problem and save the patient. The paper concludes with the description of the evaluation methodology of the Virtual Telemedicine game.

Keywords: instructional design, serious games, simulations, medical education, e-learning, virtual patients, problem-solving.

6 SERIOUS GAMES IN HEALTH-CARE

Serious games have the potential to be an important teaching tool for both formal and informal education because of their affordances of interactivity and motivation, engaging users, providing

a platform for active learning, being customized to learners, providing immediate feedback and including immersive activities. Serious games are defined as “games with an educational intent… that are engaging…have an underlying pedagogy and where learning can be implicit or explicit” (p.5)[1].

Serious games development and implementation for medical education is a growing domain. According to Graafland et al. (2012), who conducted a systematic review of serious games for medical education and surgical skills training, which included 25 research studies and covered 30 serious games published between 1995 and 2012, serious games form an innovative approach towards the education of medical professionals [2]. Across the healthcare sector, there is growing interest in improving and sustaining interaction and engagement using game technologies. Game environments provide a safe and controlled setting within which players can learn in an engaging way. Another advantage of serious games is that they allow multiple professionals to train simultaneously on one case and allow one professional to train multiple cases simultaneously. These skills are recognized as critical in reducing medical errors in dynamic high-risk environments, such as the operating room or emergency department [2]. Furthermore, serious games can provide crisis resource training, with a large variety of cases, in a relatively cheap, readily available environment that provides a viable alternative to expensive simulators. Serious games also provide training environments for disaster situations and mass casualty incidents [2].

Health games for practitioners, such as doctors and nurses, tend to be simulation-based and used for training. Simulations and serious gaming represent ideal teaching methods to optimize the knowledge and skill of residents before they are entrusted with procedures in real patients. Educators and games designers should therefore develop serious games that train professionals in order to maximize patient safety [2]. Examples of applications of simulations in the medical domain to support training and learning refer to surgical training [3], the use of online avatar-based training simulations to motivate people who exhibit signs of post-deployment stress to seek help [4], and the facilitation of surgical and endoscopic training [5]. Many of the serious games included in Graafland et al’s (2012) systematic review of serious games in medical education covered team training in acute and critical care and dealing with mass casualty incidents. Other games covered more specific areas of healthcare, such as training for coronary artery bypasses and knee joint surgery and assessing and resuscitating patients with burns.

This paper is a report on the instructional design of Virtual Telemedicine, a simulation-based serious game in health-care, created by repurposing educational content [6]. The development of the game was a small part of a larger research project, namely the mEducator Best Practice Network (www.meducator.net), an EU-funded e-learning project that sought to enable specialized state-of-the-art medical educational content to be discovered, retrieved, shared and reused across European institutions [7,8]. The paper starts with the identification of the problem to ensure that it is learning-related and therefore amenable to a solution through instructional interventions. It then proceeds to a detailed description of the instructional design of a simulation-based serious game, one of the possible solutions to the problem. The analysis of the learning context, the learners and the learning task, and the identification of goals and objectives of the proposed instructional intervention were the first steps of the instructional analysis. Following, is the suggestion of a generative, low-scaffolding strategy for problem-solving instruction, based on the assumption that the learners, in this particular case practicing doctors, have well-organized and extensive content knowledge. The main part of the paper focuses on the design and development of the prototype of the Virtual Telemedicine game and the justification of several changes that were made as part of the game’s re-design (beta-version) to increase its pedagogical value. The paper concludes with the description of the evaluation methodology of the Virtual Telemedicine game.

7 INSTRUCTIONAL ANALYSIS

7.8 Problem identification

The analysis of instructional context involves needs assessment and a description of the environment in which instruction will take place. A problem-based needs assessment was followed

in this project, whose first step was the problem determination. The identification of the problem was based on the literature strand that focused on research involving medical education and ways to make it more effective for supporting problem solving. Part of the mission of medical institutions, universities and hospitals is the training of medical students and practicing doctors for problem-solving in real-life scenarios [9]. In medical emergency situations a doctor is asked to make important decisions for the treatment of a patient within minutes or even seconds and his/her actions result in saving or losing a patient's life. Often times the doctor is not physically next to a patient and has to make crucial decisions remotely. For example the doctor may work at the emergency department of a hospital and may be asked to provide assistance to the paramedics of an ambulance that is transferring the patient. There is an identified need to train medical students for problem-solving in real-life situations of medical care through a telemedicine system. This problem is learning-related and therefore amenable to a solution through instructional interventions.

7.9 A simulation-based game as a solution

There is evidence to suggest that doctors who train medical students should consider incorporating games and simulations to their practice [10]. Therefore, a simulation-based serious game in the domain of health-care was one of possible suggested solutions to the identified problem. This solution was chosen because it can provide students with ample opportunities to practice their patient-treatment skills in a safe and controlled e-learning environment that simulates real-life conditions but avoids the risks associated with dealing with real patients. The game makes use of data from an electrocardiogram (ECG) and its overall learning goal is for practicing doctors or medical students to learn how to respond to a medical emergency situation to treat the symptoms of a virtual patient by taking specific actions and examining their results based on immediate feedback provided by both the system and the virtual patient.

7.10 Learning goal and objectives

After the learning goal of the game has been identified, the next step was to identify the type of learning outcomes that the goal represents. The goal refers to intellectual skill outcomes, which is the predominant objective of instruction in training settings, and more specifically, to domain-specific problem solving. Problem solving refers to a learned capability involving selection and application of multiple rules. Many times, learners must select from a number of possible rules, whether relational or procedural, and apply those rules in a unique sequence and combination to solve a previously unencountered problem [11]. Once learners have acquired the ability to solve problems in a specific domain, they may apply that ability to similar types of problems. An example that comes from the medical domain refers to nursing students who acquire problem solving ability when they learn to write nursing-case plans for patients who have a unique set of physical problems, medications and other treatments. Intellectual skills are hierarchical in nature. Learners must be able to make discriminations among objects before they can identify concrete concepts to use them in rules and they must have acquired the rules they will combine in unique ways to create domain-specific problem solving. In this particular case, the learner is given the history of the patient and the patient's current symptoms, and is asked to proceed to a diagnosis and treatment within a very limited period of time.

The specification of learning objectives constituted the next step of the instructional analysis. These have been identified with the help of a doctor who was the subject matter expert of the project. Examples of learning objectives are the following: "Your objective in this educational scenario is to learn how to assess a typical medical history of palpitations, diagnose one of the major causes of cardiac arrhythmias with the use of ECG monitoring and give the appropriate assistance to your patient until he arrives to the hospital. You will learn how to treat this arrhythmia with the use of appropriate drugs. You will also learn about possible complications of malpractice that arise from common mistakes in treating such a condition" (Virtual Telemedicine scenario "Palpitations").

8 A SIMULATION AS A STRATEGY FOR PROBLEM-SOLVING INSTRUCTION

An assumption made in this project is that the learners, more specifically practicing doctors, have well-organized and extensive content knowledge, good cognitive knowledge strategies, high aptitude, high motivation and a sufficient amount of time to interact with the game. Therefore a more generative, low-scaffolding instructional strategy was considered as more appropriate.

A simulation was used as a macrostrategy for problem-solving instruction. A simulation is defined as “an activity that attempts to mimic the most essential features of reality but allows learners to make decisions within this reality without actually suffering the consequences of their decisions” (p.230) [11]. Simulations are considered as “facilitators of virtual experiences” (p.11) [3] and are associated with exploratory learning that in this case helps learners rehearse skills for the real-world. Instructional simulations generally provide a problem situation through depiction of a system in operating form and then require the learner to interact with the problem. With every action of the learner, there is a response within the simulation, which can be lifelike and immediate.

Simulations have several advantages for learning, such as the following: they portray a meaningful context, they can be quite complex, they expose learners to alternative solutions, they require problem solving in situations in which there is no single correct answer, they allow learners to see the consequences of their solutions and they require learners to predict the effects of their actions [11]. The Virtual Telemedicine serious game is simulation-based in the sense that it implements “rigorously structured scenarios with a highly refined set of rules, challenges, and strategies which are carefully designed to develop specific competencies that can be directly transferred into the real world” (p.17) [1]. It is important to note that the game not only allows users to practice in the context of given scenarios but also allows them to write scenarios of their own with no prior knowledge of programming required.

9 DESIGN AND DEVELOPMENT OF THE PROTOTYPE OF VIRTUAL TELEMEDICINE

4.1. Description of prototype version

The prototype version of the game included one scenario, called “Precordial pain” and had the interface shown in Figure 1. The user is informed that the patient is facing a problem, an acute coronary syndrome. To take action and save the patient the user can choose one of seven possible options: provide aspirine to the patient, or provide alternative drugs such as Atropine, Morphine, etc. Upon successful completion of the game, medical students learn that in this specific scenario where a patient suffers from precordial pain, sensible choices are Aspirine, Morphine, Nitroglycerin Sublingual, Metoprolol and Oxygen. Making the wrong choice may bring negative side-effects. After giving the wrong medication, the patient might lose his conscience because of cardiac arrest. In this case, the user is provided with different options that correctly correspond to this new situation. Namely, the user is asked to give a defibrillation treatment. Failure to do so will lead to the death of the patient. The scenario is won after the patient is stabilized and the user gives him the correct combination of drugs, the appropriate in an acute coronary syndrome.

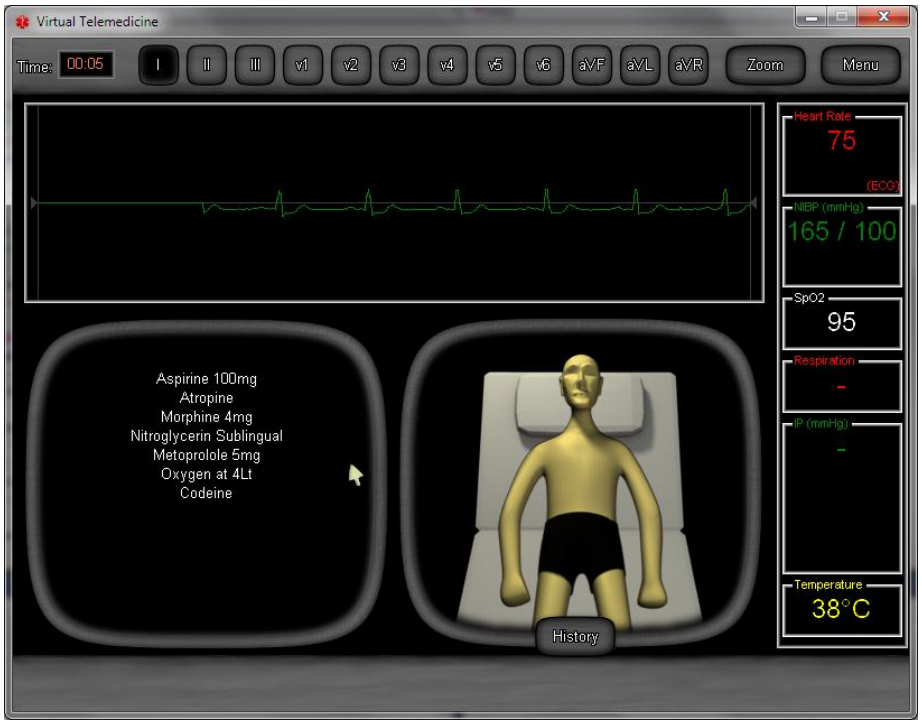


Fig. 1: Screencapture showing the graphical user interface of the prototype of the Virtual Telemedicine game

4.2. Technical specifications of the game

With regard to the technical specifications, the virtual telemedicine game implements a scripting language called Scribulance, which is a blend of C and Pascal. Scribulance is a user-friendly high-level scripting language developed specifically for the purpose of the game. It enables the creation of custom scenarios, which are currently text-based. The game was created using the programming language C# with XNA for the game. In order to support a high level scripting language for developing the game's scenarios a scenario compiler was also created using C with Flex/Bison for the compiler and C# to import the scenarios into the game. For testing and evaluation purposes, several scripts that used the features of the scripting language were implemented to examine whether they worked properly and a detailed command-by-command debugging was conducted to examine that the flow of the code worked as intended.

4.3. Access to the game

"Virtual Telemedicine" is currently working as a standalone application on PCs with the Windows operating system. The game can be shared and repurposed (through changing its scenarios, or creating new ones and compiling them with the included Virtual Telemedicine compiler). It is discoverable through different instantiations of mEducator, such as mEducator 2.0 and mEducator 3.0. Examples of such instantiations are the following:

a) an instantiation that utilizes Moodle [12,13],

b) an instantiation that utilizes the Drupal CMS,

c) an instantiation that utilizes the Linked Labyrinth + , a platform for Virtual patients,

d) Metamorphosis+, an instantiation that is based on Elgg and

e) Melina+, an extended version of Drupal 7, which is offered as an installation profile and enables web site administrators to install a learning management system, focused on medical education.

10 DESIGN AND DEVELOPMENT OF THE BETA-VERSION OF VIRTUAL TELEMEDICINE

The game was evaluated through the instructional design paradigm and changes were made as part of the game's redesign and development to increase its pedagogical value. Apart from scenario
"Precordial pain", two scenarios have been added to the beta version of the game: a) "Palpitations", in which the patient experiences a cardiac arrhythmia and b) "Syncope", in which the patient is diagnosed with an atrioventricular block.

Some of the changes that were made to the beta-version of the game included the addition of the overall goal of the game, the objectives and context for each scenario, the patient's history, as well as specific instructions to scaffold the user in making a choice among several treatment options. These changes are depicted in Fig.2 and are provided to the user when he makes the choice of one of the three possible scenarios (Palpitations, Precordial Pain and Syncope) in the main menu.

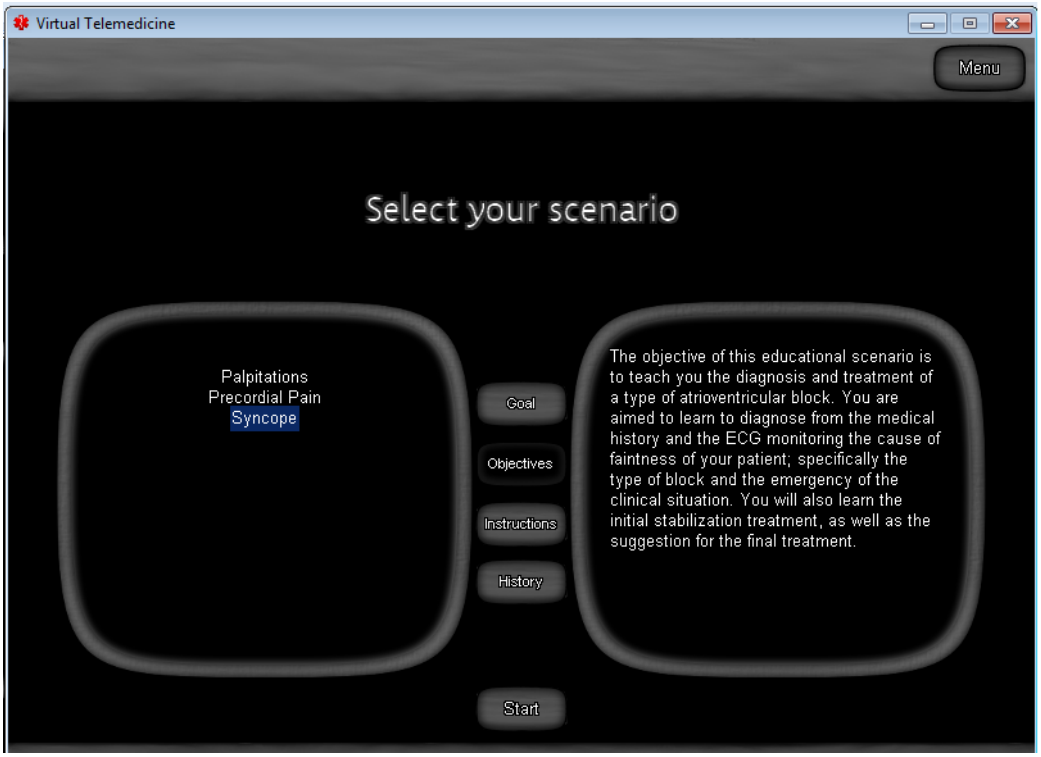


Fig. 2: Screencapture showing the objectives of the Virtual Telemedicine game (beta version)

However, these options (Goal, Objectives, Instructions, History) are always available to the user, as can be seen in Figure 3, as he/she may need to refer to them while playing the game.

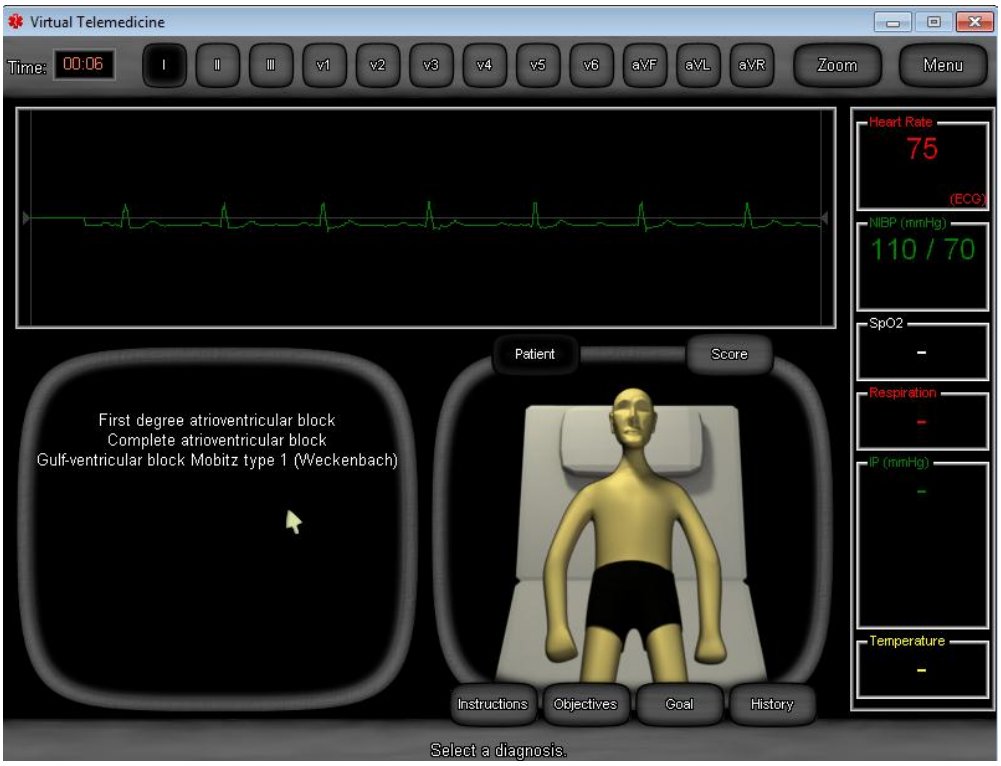


Fig. 3: Screencapture showing the Syncope scenario of the Virtual Telemedicine game (beta version)

An example of objectives as these are provided in the Syncope scenario, which is the one selected in Fig. 2 and Fig. 3 is the following:

“The objective of this educational scenario is to teach you the diagnosis and treatment of a patient with a type of atrioventricular block. You are aimed to learn to diagnose from the medical history and the ECG monitoring the cause of faintness of your patient; specifically the type of block and the emergency of the clinical situation. You will also learn the initial stabilization treatment, as well as the suggestion for the final treatment”.

The instructions put the game in context. An example of instructions provided in the same scenario is the following:

“You are a doctor in the emergency department of the local hospital and you have to provide your assistance to the paramedics of an ambulance that is transferring a 68-year-old patient who had lost consciousness at home. You have to keep him hemodynamically stable, by administering the right drugs after the right diagnostic evaluation of his ECG. You have to have your patient safely transferred to the cardiology department of your hospital and suggest the correct final treatment”.

An example of the patient’s history is the following:

“The patient is 68-years-old. We know he used to smoke but the rest of his history is unknown. He lost consciousness and fainted in his house”.

Another important change made as part of the redesign of the game refers to instant feedback provided to the user when he/she interacts with the game. The user receives a clearly visible positive or negative score for every one of his actions while trying to solve the problem of the game. In addition, the user can track his score while playing the game. In the example provided in Fig. 3, the user is asked to select a diagnosis and gets immediate feedback in the form of points. A positive figure in green colour (e.g. +15) briefly appears on top of the correct selection or a negative figure in red colour (e.g. -10) briefly appears on top of an incorrect selection.

Finally, a detailed report is provided at the end of the game outlining the user's choices, clearly indicating which of them were successful and including the total amount of time he/she required to solve the problem and save the patient (Fig. 4).

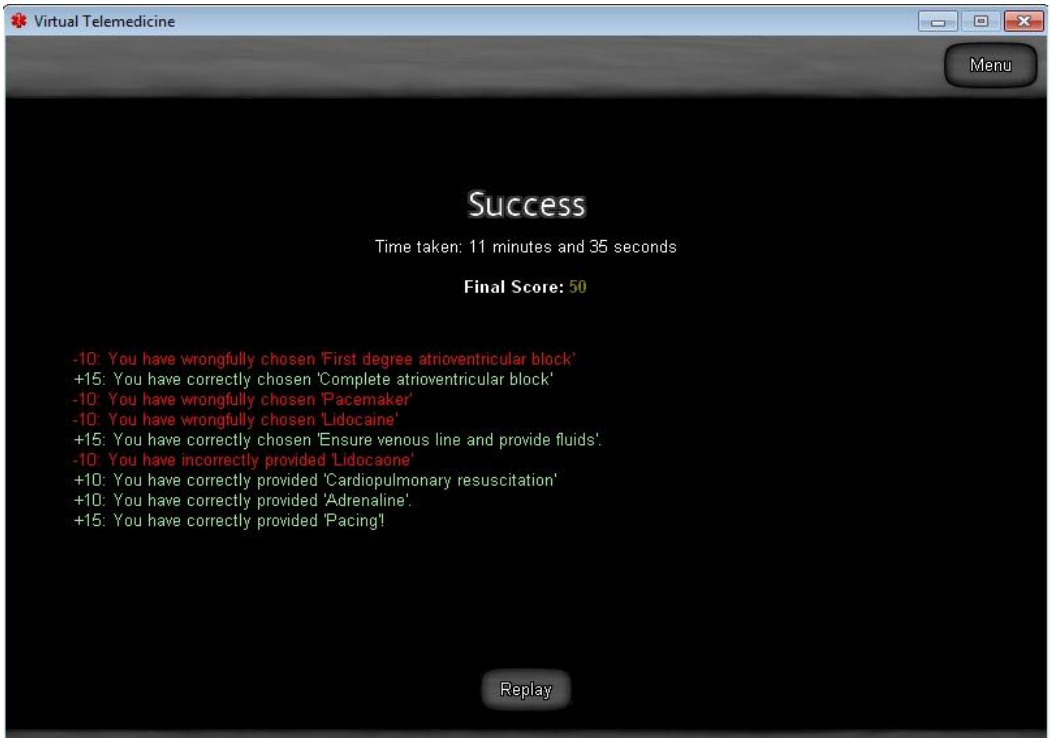


Fig. 4: Screencapture showing the final score and the time needed to complete the syncope scenario of the Virtual Telemedicine game (beta version)

11 USABILITY TESTING METHODOLOGY

The proposed evaluation paradigm for Virtual Telemedicine is usability testing. Usability testing involves measuring typical users' performance on carefully planned tasks that are typical of those for which the system was designed [14]. Tests often take place in laboratory-like conditions that are controlled. Users' performance is then quantified and analyzed. User satisfaction data from questionnaires tends to be categorized and average ratings are presented. Usability testing can be combined with observations, recording the users using video and logging their interactions with the software.

7.11 Participants

The target population for the evaluation of the Virtual Telemedicine game will be resident doctors. The sample of this study will consist of approximately 20 residents from different parts of Greece (convenience sample). Due to specific time limitations and the resident doctors' limited availability, as well as their geographical disparity, as doctors live and work in different cities in Greece, usability testing in a laboratory setting will not be feasible. Detailed step-by-step instructions including screencaptures will be sent to doctors so that they can install the game on their personal computers and try it out at their convenience. The participants will then be asked to complete an online questionnaire and send a log file containing a report of their actions, that is automatically created by the system, to the researchers.

6.2 Usability evaluation instrument

The usability evaluation instrument will be a questionnaire specifically developed for this game, which consists of four parts. The first part refers to participants' demographical information including sex, age, years of experience in medical education, area of medical expertise, and previous experience with games, serious games and serious games in the medical domain. The second part asks participants to install the game, try to solve the problems of the three scenarios and then rate the difficulty level for performing the following actions:

1. Install the game
2. Select one of the proposed medication/treatments for the patient
3. Keep track of their score while playing the game
4. Access the game objectives
5. Access the game instructions
6. Access the patient history
7. Solve the problem

A four-point Likert scale is used for the rating of the actions above, with the following options: a) Easy, b) Ok, c) Difficult, d) Needed help.

The third part of the questionnaire includes 21 items focusing on the evaluation of the game. Participants are asked to rate the statements presented below using a four-point Likert scale that includes the following options: a) Strongly Disagree, b) Disagree, c) Agree, d) Strongly Agree and e) I don't know.

The items are the following:

1. The game will be interesting for medical students.
2. The game is useful for medical student training.
3. The game is user-friendly.
4. The game provides ways to recover after making a mistake.
5. I like the interface of the game.
6. The game graphics are adequate.
7. The terminology used is correct.
8. The terminology used is consistent.
9. The response time of the game is as expected.
10. The feedback I receive when I make a choice is adequate.
11. The feedback I receive when I make a choice is confusing.
12. I can learn from my mistakes when I play the game.
13. The game is not challenging for me.
14. I feel "in control" when I play the game.
15. The time allowed by the game for the doctor to save the patient is sufficient.
16. I needed more time to be able to solve the problem in Scenario Precordial Pain.
17. I needed more time to be able to solve the problem in Scenario Syncope.
18. I needed more time to be able to solve the problem in Scenario Palpitations.

19. If I were an instructor I would like to use the game in a classroom setting with my students.

20. I would recommend the game to my colleagues.

21. The game is complicated.

The fourth part of the questionnaire consists of five open-ended questions, which are the following:

1. What did you like about the Virtual Telemedicine serious game?
2. What did you not like about the Virtual Telemedicine serious game?
3. What did you find confusing or difficult to use in Virtual Telemedicine?
4. How would you suggest improving Virtual Telemedicine?
5. If you were training medical students, would you be interested in using this game in your class?

12 NEXT STEPS: IMPLEMENTATION AND SUMMATIVE EVALUATION

One of the next steps of this project refers to the implementation of Virtual Telemedicine, the use of the simulation in the context for which it was intended. This would allow the examination of whether learners are actually learning the concepts that are part of the simulation and whether they are indeed in a position to solve problems with respect to medical emergency situations. This is in accordance with guidelines provided by Graafland et al. (2012) who pointed out that games need to be designed to fit into residency teaching programs if they are to be used as a way of preventing medical errors [2].

Without an implementation phase, it is quite possible for learners to be successful within the simulation without necessarily acquiring the ability to correctly apply the principles in other cases. Their behavior within the simulation may be trial-and-error or based on faulty, albeit successful reasoning. During a debriefing session that will be designed when Virtual Telemedicine is used in an instructional context as part of its implementation plan, learners will be required to explain their understanding of the given state, the goal state, their selection of principles to solve the problem and how their actions moved the situation from the given state to the goal state.

With regard to summative evaluation the next variable that will be measured is transfer, the application of new learning to other situations, otherwise referred to as learning outcomes, which in the context of the present research study will measure “near transfer”, such as the ability to solve similar problems in similar situations.

With regard to future research, as Graafland et al.(2012) pointed out, games developed or used to train medical professionals need to be validated before they are integrated into teaching methods. Therefore further research should define valid performance parameters and formally validate programs before serious games can be seen as fully fledged teaching instruments for medical and surgical professionals [2].

REFERENCES

[1] Ulicsak, M. & Wright, M. (2010). Games in education: Serious Games. Bristol, Futurelab. Retrieved from: http://media.futurelab.org.uk/resources/documents/lit_reviews/Serious-Games_Review.pdf

[2] Graafland, M., Schraagen, J.M. & Schijven, M.P. (2012). Systematic review of serious games for medical education and surgical skills training. British Journal of Surgery. 99, pp1322-1330. DOI: 10.1002/bjs.8819

[3] De Freitas,S.(2006). Learning in immersive worlds. A review of game-based learning. Retrieved from: http://www.jisc.ac.uk/media/documents/programmes/elearninginnovation/gamingreport_v3.pdf

[4] Albright, G, Goldman, R., Shockley,K., McDevitt, F. & Akabas, S. (2012). Using an Avatar-Based Simulation to Train Families to Motivate Veterans with Post-Deployment Stress to Seek Help at the VA. Games for Health Journal, 1(1), 21-28. doi:10.1089/g4h.2011.0003.

[5] Enochsson, L., Isaksson, B., Tour, R., Kjellin, A., Hedman, L., Wredmark, T. & Tsai-Fellander, L. (2004). Visuospatial skills and computer game experience influence the performance of virtual endoscopy. Journal of Gastrointestinal Surgery, 8(7), 874–880.

[6] Mougiakakou, S., Kyriacou, E., Perakis, K., Papadopoulos, H., Androulidakis, A., Konnis, G., Tranfaglia, R., ecchia, L., Bracale, U., Pattichis, C., & Koutsouris, D., (2011). Towards the Integrated Provision of eHealth Tools and Services: INTERMED and the Experience in the Region of South-East Mediterranean, BioMedical Engineering OnLine, 10 (49).

[7] Nicolaidou, I., Bamidis, P., Antoniades, A., Nikolaidou, M., Pattichis, C. & Giordano, D. (2011). mEducator: Multi-type content sharing and repurposing in medical education. In Proceedings of World Conference on Educational Multimedia, Hypermedia and Telecommunications 2011 (pp. 3740-3745). Chesapeake, VA: AACE.

[8] Bamidis, P., Kaldoudi, E., & Pattichis, C. (2009). mEducator: A Best Practice Network for Repurposing and Sharing Medical Educational Multi-type Content. Leveraging Knowledge for Innovation in Collaborative Networks. IFIP Advances in Information and Communication Technology, 307, 769-776.

[9] Issenberg B., McGaghie W., Hart I., Mayer J., Felner J., Petrusa E., Waugh R., Brown D., Safford R., Gessner I., Gordon D. & Ewy G. (1999). Simulation Technology for Healthcare Professional Skills Training and Assessment. JAMA, 282(9), 861-866.

[10] Schwaab, J, Kman, N., Nagel, R., Bahner, D., Martin, D., Khandelwal, S.,…Nelson, R. (2011). Using Second Life Virtual Simulation Environment for Mock Oral Emergency Medicine Examination. Society for Academic Emergency Medicine, 18(5), 559-562.

[11] Smith, P. & Ragan, T. (2005) Instructional Design. Wiley. 3rd edition.

[12] Bamidis, P., Konstantinidis, S., Bratsas, C., & Iyengar, S. (2011). Federating Learning Management Systems for Medical Education: a persuasive technologies perspective, In Proc. of Computer Based Medical Systems (CBMS) 2011, IEEE 2011.

[13] Konstantinidis, S., Lakka, C., Bratsas, C., Pappas, C. & Bamidis, P. (2011). Semantic description of digital educational objects in an e-learning environment, In Proc. Of the 2nd Panhellenic Conference on the introduction and use of ICT in the educational process, pp. 413-422.

[14] Preece, J., Rogers, Y. & Sharp, H. (2002). Interaction design, beyond human-computer interaction. John Wiley & Sons, Inc.