ΠΑΝΕΠΙΣΤΗΜΙΟ ΜΑΚΕΔΟΝΙΑΣ
ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ
ΤΜΗΜΑΤΟΣ ΕΦΑΡΜΟΣΜΕΝΗΣ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΣΧΕΔΙΑΣΜΟΣ ΚΑΙ ΑΝΑΠΤΥΞΗ ΠΑΙΧΝΙΔΙΟΥ ΣΟΒΑΡΟΥ ΣΚΟΠΟΥ ΜΕ C++ ΚΑΙ OPENGL ΓΙΑ ΕΚΜΑΘΗΣΗ ΤΗΣ ΓΛΩΣΣΑΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ C++

Διπλωματική Εργασία

του

Ελευθεριάδη Σάββα

Θεσσαλονίκη, Ιούνιος 2020

ΣΧΕΔΙΑΣΜΟΣ ΚΑΙ ΑΝΑΠΤΥΞΗ ΠΑΙΧΝΙΔΙΟΥ ΣΟΒΑΡΟΥ ΣΚΟΠΟΥ ΜΕ C++ ΚΑΙ OPENGL ΓΙΑ ΕΚΜΑΘΗΣΗ ΤΗΣ ΓΛΩΣΣΑΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ C++

Ελευθεριάδης Σάββας

Πτυχίο Εφαρμογών Πληροφορικής στη Διοίκηση και στην Οικονομία,
ΤΕΙ Ιονίων Νήσων, 2008

Διπλωματική Εργασία

υποβαλλόμενη για τη μερική εκπλήρωση των απαιτήσεων του

ΜΕΤΑΠΤΥΧΙΑΚΟΥ ΤΙΤΛΟΥ ΣΠΟΥΔΩΝ ΣΤΗΝ ΕΦΑΡΜΟΣΜΕΝΗ ΠΛΗΡΟΦΟΡΙΚΗ

Επιβλέπων Καθηγητής
Ξυνόγαλος Στυλιανός

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την ……...

ΞΥΝΟΓΑΛΟΣ ΣΤΥΛΙΑΝΟΣ	ΣΑΤΡΑΤΖΕΜΗ ΜΑΡΙΑ - ΑΙΚΑΤΕΡΙΝΗ	ΧΑΤΖΗΓΕΩΡΓΙΟΥ ΑΛΕΞΑΝΔΡΟΣ
.....	.....	.....

Ελευθεριάδης Σάββας

.....

Περίληψη

Η διπλωματική εργασία αφορά την ανάπτυξη παιχνιδιού σοβαρού σκοπού με την γλώσσα προγραμματισμού C++ και το API γραφικών OpenGL, για την εκμάθηση βασικών και προχωρημένων εννοιών της γλώσσας προγραμματισμού C++. Το παιχνίδι σχεδιάστηκε ώστε να είναι ανεξάρτητο πλατφόρμας (cross-platform). Η σχεδίαση του πραγματοποιήθηκε με το πλαίσιο σχεδίασης EFM, ενώ η αξιολόγηση του έγινε με το πλαίσιο αξιολόγησης MEEGA.

Βασικός στόχος του παιχνιδιού είναι να εισαγάγει με διασκεδαστικό, όσο και σοβαρό, τρόπο τους χρήστες στην γλώσσα προγραμματισμού C++. Η ιδιαιτερότητα της εφαρμογής αυτής εντοπίζεται 1) στη μέθοδο εκπαίδευσης των χρηστών και 2) στην επιλογή των εργαλείων που χρησιμοποιήθηκαν για την δημιουργία της εφαρμογής.

Οι περισσότερες εφαρμογές εκπαιδευτικού σκοπού αυτού του είδους (εκπαίδευση σε μια γλώσσα προγραμματισμού) παρουσιάζουν τις τεχνικές λεπτομέρειες μιας γλώσσας μέσω ενός ενσωματωμένου “παιχνιδιού”, το οποίο χρησιμοποιείται ως επιπλέον κίνητρο ώστε να παρακινεί τον παίχτη / εκπαιδευόμενο να συνεχίσει την εκπαίδευση του.

Με την παρούσα εφαρμογή σοβαρού σκοπού, εφαρμόζεται μια άλλη μέθοδος εκπαίδευσης του χρήστη, χρησιμοποιώντας τεχνικές και μηχανισμούς που τελικό σκοπό έχουν να διεγείρουν το αίσθημα σοβαρότητας και υπευθυνότητας του εκπαιδευομένου κατά την εκμάθηση της γλώσσας και όχι απλά να τον διασκεδάσουν κατά την εκπαίδευση του, διεγείροντας απλά το αίσθημα της ψυχαγωγίας.

Οι κύριοι άξονες που χρησιμοποιούνται για να επιτευχθεί αυτό είναι 1) ρεαλισμός και 2) υπευθυνότητα, συμπεριλαμβανομένων και των ιδιοτήτων του πλαισίου σχεδίασης. Τέλος, η εφαρμογή δημιουργήθηκε σε χαμηλό επίπεδο, μόνο με τη χρήση της γλώσσας C++, του API γραφικών OpenGL, και κάποιων επιπρόσθετων βιβλιοθηκών λογισμικού χαμηλού επιπέδου.

Λέξεις Κλειδιά: παιχνίδι σοβαρού σκοπού, προγραμματισμός, c++, opengl, πλαίσιο σχεδίασης, πλαίσιο αξιολόγησης, cross-platform

Abstract

This thesis focuses on developing a serious game with the C++ programming language and the OpenGL graphics API to learn the basic and advanced concepts of the C++ programming language. The game was designed to be cross-platform. It's design was done with the EFM design framework while its evaluation was done with the MEEGA evaluation framework.

The main purpose of the game is to introduce users to the C++ programming language in a fun, as well as serious, way. The difference of this application has to do with 1) the training method of users and 2) the selection of tools used to create the application.

Most educational applications of this kind (training in a programming language) present the technical details of a language through a built-in "game", which is used as an additional motivation to the player / learner to continue their training.

With this serious game, another method of educating the user is applied, using techniques and mechanisms, whose ultimate goal is to stimulate the learner's sense of seriousness and responsibility when learning the language and not just to entertain him during his training, simply by stimulating the feeling of entertainment.

The main axes used to achieve this are 1) realism and 2) responsibility, including the properties of the design framework. Finally, the application was created on a low level, using only the C++ language, the OpenGL graphics API and some additional low-level software libraries.

Keywords: serious game, programming, c++, opengl, design framework, evaluation framework, cross-platform

Πρόλογος – Ευχαριστίες

Η παρούσα εργασία μου έδωσε την δυνατότητα να ερευνήσω τον χώρο των παιχνιδιών σοβαρού σκοπού και συγκεκριμένα ενός τομέα που με ενδιαφέρει πολύ, αυτόν της εκμάθησης μιας προγραμματιστικής γλώσσας.

Θα ήθελα να ευχαριστήσω προσωπικά τον καθηγητή μου κ. Ξυνόγαλο Στέλιο που μου έδωσε την ευκαιρία να το κάνω. Το μάθημα του ήταν ο κυριότερος λόγος που επέλεξα το μεταπτυχιακό αυτό πρόγραμμα καθώς συνδύαζε δυο (2) από τις πιο αγαπημένες μου ασχολίες στον υπολογιστή, παιχνίδια και προγραμματισμό. Τον ευχαριστώ επίσης για την υπομονή του, καθώς και για τις πολλές, γρήγορες & ακριβείς διορθωτικές προτάσεις του ώστε η εργασία να έχει αυτή τη μορφή.

Ευχαριστώ κάποιους ανθρώπους που έδειξαν ενδιαφέρον κατά την ανάπτυξη της εργασίας αυτής και της προσπάθειας που έκανα γενικότερα κατά την ολοκλήρωση αυτής της διαδρομής.

Τέλος, ευχαριστώ όσους ξόδεψαν λίγο από τον χρόνο τους, δοκιμάζοντας και αξιολογώντας την εφαρμογή. Η βοήθεια τους θα αποδειχτεί πολύτιμη για την μελλοντική βελτίωση της εφαρμογής.

Περιεχόμενα

Εικόνες

Πίνακες

Πηγαίοι Κώδικες

1 Εισαγωγή

1.1 Πρόβλημα – Σημαντικότητα του θέματος

Το θέμα της διπλωματικής εργασίας αφορά στη σχεδίαση και ανάπτυξη μιας εφαρμογής παιχνιδιού σοβαρού σκοπού, για την εκμάθηση της γλώσσας προγραμματισμού C++ σε περιβάλλον Windows. Το περιβάλλον των Windows επιλέχθηκε λόγω της φύσης της εφαρμογής (απαιτεί αρκετή πληκτρολόγηση) καθώς και για λόγους εύκολης πρόσβασης και χρήσης από όλους.

Ωστόσο, ο πηγαίος κώδικας της εφαρμογής είναι ανεξάρτητος πλατφόρμας (cross-platform), κυρίως λόγω της χρήσης OpenGL αλλά και της γλώσσας προγραμματισμού C++, ένας συνδυασμός ο οποίος υποστηρίζεται από τη συντριπτική πλειοψηφία των λειτουργικών συστημάτων και συσκευών.

Η OpenGL είναι μια διασύνδεση προγραμματισμού εφαρμογών (Application Programming Interface - API) γραφικών, η οποία πρακτικά “συνδέει” τα προγράμματα των χρηστών με τις 3D κάρτες γραφικών ώστε οι προγραμματιστές να μπορούν να δημιουργήσουν εφαρμογές γραφικών 2D/3D μεγάλης απόδοσης που θα εκτελούνται σε όλα τα συστήματα και σε όλες τις κάρτες γραφικών.

Οι λόγοι για τους οποίους επιλέχθηκε το συγκεκριμένο API γραφικών και όχι κάποιο άλλο π.χ. DirectX αλλά και γιατί γενικά επιλέξαμε ένα τρίτο API γραφικών και δεν χρησιμοποιήσαμε δικό μας κώδικα για τα γραφικά ή κάποια μηχανή παιχνιδιών, θα αναλυθούν εκτενέστερα στα επόμενα κεφάλαια. Η γλώσσα προγραμματισμού C++ επιλέχθηκε για την ανάπτυξη της εφαρμογής για τους παρακάτω λόγους:

- ✔ **Έλεγχος:** Είναι γνωστό ότι η γλώσσα προγραμματισμού C++ είναι μια από τις πιο ισχυρές γλώσσες, κυρίως γιατί δίνει μεγάλο έλεγχο στον προγραμματιστή για τον έλεγχο των πόρων της συσκευής.

- ✔ **Απόδοση:** Τα παραγόμενα εκτελέσιμα αρχεία είναι γραμμένα απευθείας σε γλώσσα μηχανής και έτσι εκτελούνται με τον ταχύτερο δυνατό τρόπο. Τα παιχνίδια 2D/3D απαιτούν μεγάλη ταχύτητα απόδοσης και για αυτόν το λόγο κυρίως επιλέγεται η C++ για την ανάπτυξη παιχνιδιών κάθε είδους ή πλατφόρμας.

Τέλος, η γλώσσα προγραμματισμού C++ επιλέχθηκε για την εκμάθηση των χρηστών για τους παρακάτω λόγους:

- ✔ **Τεχνολογία:** Στην εποχή μας, η αυξημένη τεχνολογία οδηγεί σε ανάγκη χρήσης δυναμικών, αντικειμενοστρεφών γλωσσών προγραμματισμού, οι οποίες υποστηρίζονται παντού και αναπτύσσονται σταθερά όπως η C++ και η Java.

- ✔ **Χρήση:** Η γλώσσα C++ κατατάσσεται ψηλά στη χρήση παγκοσμίως, κάτι που δεν θα αλλάξει σύντομα, δεδομένου ότι χρησιμοποιείται εδώ και 35 χρόνια και εξελίσσεται συνεχώς με νέες εκδόσεις π.χ. *C++17* (2017).

- ✔ **Παιχνίδια:** Βρισκόμαστε (ξανά!) στη χρυσή εποχή των παιχνιδιών. Στο παρελθόν, μόνο ένας μικρός αριθμός εταιρειών έβγαζε παιχνίδια. Σήμερα, υπάρχουν πλατφόρμες όπως το Steam, το itch.io και το Gamejolt οι οποίες χρησιμοποιούνται για εμπόριο παιχνιδιών για όλες τις συσκευές, ενώ τα παιχνίδια αναπτύσσονται ακόμα και από απλούς χομπίστες. Η γλώσσα προγραμματισμού C++ επιλέγεται από πολλούς δημιουργούς, για τους λόγους που προαναφέρθηκαν.

- ✔ **Δυσκολία:** Ένας από τους λόγους που υπάρχει έλλειψη σε εφαρμογές σοβαρού σκοπού για εκμάθηση της γλώσσας C++ είναι η δυσκολία της. Πολλές λειτουργίες της γλώσσας είναι δύσκολο να αναπαραχθούν οπτικά και λειτουργικά, τόσο για κατανόηση και εκμάθηση, όσο και συντακτικό έλεγχο, και έτσι πολλές εφαρμογές καταπιάνονται με άλλες γλώσσες.

1.2 Σκοπός – Στόχοι

Ο κύριος σκοπός της εργασίας είναι η σχεδίαση και υλοποίηση ενός παιχνιδιού, χρησιμοποιώντας απλά μια γλώσσα προγραμματισμού, στην περίπτωση αυτή την γλώσσα C++, καθώς και το δημοφιλές API γραφικών OpenGL. Αυτό σημαίνει ότι δεν θα χρησιμοποιηθεί κάποια έτοιμη μηχανή παιχνιδιών του εμπορίου όπως π.χ. Unity ή το Gamemaker Studio.

Αντιθέτως, κατά τη ανάπτυξη της εφαρμογής, θα δημιουργηθεί (σταδιακά) ένα API, το οποίο θα χρησιμοποιείται για τη διαχείριση των γραφικών, του ήχου, της βάσης δεδομένων (αν απαιτηθεί) κλπ. και γενικότερα ότι θα απαιτηθεί, αναλύοντας παράλληλα την λογική για κάθε τμήμα κώδικα που παρουσιάζεται.

Στη συνέχεια, θα ενσωματωθεί το εκπαιδευτικό περιεχόμενο στο παιχνίδι, οι μηχανισμοί εκπαίδευσης του χρήστη και οτιδήποτε περιλαμβάνεται σε ένα παιχνίδι σοβαρού σκοπού, χρησιμοποιώντας το επιλεγμένο πλαίσιο σχεδίασης.

Οι κύριοι λόγοι για τους οποίους δεν επιλέχθηκε μια έτοιμη λύση, όπως η μηχανή παιχνιδιών Unity που ενσωματώνει ήδη όλες τις δυνατότητες αναπαραγωγής (π.χ. γραφικών, ήχου κλπ.) και είναι ήδη cross-platform, είναι οι παρακάτω:

- ✔ **Έλεγχος:** Μια εμπορική μηχανή παιχνιδιών έχει πολλές έτοιμες δυνατότητες και έχει λύσει αρκετά προβλήματα κατά την ανάπτυξη της. Ωστόσο, πολλές φορές είναι περιορισμένη στα χαρακτηριστικά της, είναι αργή ή εμφανίζει προβλήματα (bugs) τα οποία δεν μπορούν να λυθούν ή να υλοποιηθούν εναλλακτικές λύσεις λόγω έλλειψης εμπειρίας ή χρόνου.

- ✔ **Εκμάθηση:** Μια εμπορική μηχανή παιχνιδιών απαιτεί αρκετό χρόνο για να τη μάθει κάποιος, προκειμένου να είναι έτοιμος να δημιουργήσει κάτι με αυτήν. Κύριος σκοπός της εργασίας, ωστόσο, δεν είναι η εκμάθηση μιας τέτοιας μηχανής αλλά η δημιουργία ενός παιχνιδιού σοβαρού σκοπού. Η εκμάθηση μιας μηχανής παιχνιδιών απαιτεί αρκετό χρόνο, ενώ ένας δευτερεύων στόχος ήταν και η παράλληλη παρουσίαση τεχνικών θεμάτων χαμηλού επιπέδου στον προγραμματισμό παιχνιδιών και όχι η επικέντρωση σε μια συγκεκριμένη μηχανή παιχνιδιών.

- ✔ **Εμπειρία:** Η εκμάθηση της γλώσσας C++ δεν επιλέχθηκε τυχαία. Λόγω εμπειρίας στη συγκεκριμένη γλώσσα προγραμματισμού, στη δημιουργία μηχανής παιχνιδιών αλλά και στη δημιουργία παιχνιδιών γενικότερα, λήφθηκε από τον συγγραφέα της διπλωματικής η απόφαση υλοποίησης του παιχνιδιού, αξιοποιώντας τις υπάρχουσες γνώσεις του, περιορίζοντας τους αναγνώστες στη χρήση μιας γλώσσας προγραμματισμού και όχι σε κάποια (από τις πολλές) μηχανές παιχνιδιών.

- ✔ **Γνώση:** Η γνώση που θα αποκομίσουν οι αναγνώστες από την παρούσα εργασία θα τους επιτρέψει να αναπτύξουν τα δικά τους παιχνίδια, σοβαρού σκοπού και μη, ξεκινώντας από δυο (2) εκ των δημοφιλέστερων εργαλείων στην κατασκευή παιχνιδιών, της C++ και της OpenGL.

Επίσης, οι κύριοι λόγοι για τους οποίους επιλέχθηκε η χρήση του OpenGL API για τη διαχείριση γραφικών αλλά και γιατί δεν επιλέχθηκε ένα άλλο π.χ. DirectX των Windows, είναι οι παρακάτω:

- ✔ **Ανεξάρτητο πλατφόρμας:** Το OpenGL σχεδιάστηκε εξ' αρχής να είναι ανεξάρτητο πλατφόρμας (cross-platform). Έτσι, κώδικας που χρησιμοποιεί OpenGL μπορεί να μεταγλωττιστεί σε ένα πλήθος άλλων συστημάτων και συσκευών, κινητών και μη και να εκτελεστεί απρόσκοπτα. Το DirectX είναι ένα αντίστοιχο API γραφικών (και όχι μόνο), το οποίο όμως λειτουργεί μόνο σε περιβάλλον Windows.

- ✔ **Υποστήριξη:** Η πρώτη έκδοση του OpenGL v1.0 παρουσιάστηκε το μακρινό 1992. Από τότε έχει δεχθεί πολλές αναβαθμίσεις, με την τελευταία έκδοση του να είναι η v4.6 και να υποστηρίζεται από όλους τους κατασκευαστές καρτών γραφικών σήμερα.

1.3 Συνεισφορά

Η παρούσα εργασία αφορά τον σχεδιασμό και υλοποίηση μιας εφαρμογής σοβαρού σκοπού στην εκπαίδευση και συγκεκριμένα στην εκμάθηση μιας γλώσσας προγραμματισμού. Υπάρχουν αρκετές εφαρμογές που αναλαμβάνουν να εκπαιδεύσουν τους χρήστες σε μια γλώσσα προγραμματισμού. Κάποιες εφαρμογές το κάνουν πετυχημένα, άλλες όχι και τόσο.

Η παρούσα εφαρμογή σχεδιάστηκε ώστε να καλύψει τα κενά που διαπιστώθηκαν σε κάποιες εκπαιδευτικές εφαρμογές κατά την χρήση τους (οι οποίες και θα παρουσιαστούν παρακάτω) και έχουν να κάνουν πρωτίστως με τη μεθοδολογία που επιλέγεται από την πλειονότητα των εφαρμογών αυτών ώστε να "εθίσουν" τους παίχτες να συνεχίσουν την ενασχόληση με τα παιχνίδια αυτά.

Αρκετές από τις εφαρμογές αυτές χρησιμοποιούν ως κινητήριο “όχημα” την διασκέδαση ώστε να μην κουράζουν τους παίχτες κατά την εκπαίδευση τους. Τι σημαίνει όμως “διασκέδαση”; Είναι ένας όρος αρκετά αφηρημένος που δεν μπορεί να μετρηθεί εύκολα, (πόσο μάλλον να σχεδιαστεί, να υλοποιηθεί και τελικά να παρουσιαστεί στην οθόνη) ενώ μεταβάλλεται σε μεγάλο βαθμό από πολλούς παράγοντες π.χ. περιεχόμενο, ηλικία, επάγγελμα, ψυχολογία, κίνητρο.

Σε κάποιες περιπτώσεις μάλιστα το είδος της “διασκέδασης” σε ένα παιχνίδι μπορεί να απωθεί συγκεκριμένες ομάδες χρηστών επειδή απλά οι ομάδες αυτές δεν μπορούν να ταυτιστούν με το είδος της διασκέδασης (που υποθετικά έρχεται να βοηθήσει αντί να απωθήσει).

Με τη παρούσα εργασία, επιχειρείται ο αρμονικός συνδυασμός του εκπαιδευτικού περιεχομένου με αρκετά ρεαλιστικά στοιχεία, τα οποία θα προκαλούν το ενδιαφέρον και θα παρέχουν κίνητρο στο χρήστη να προχωρήσει, ενώ θα ενισχύουν ταυτόχρονα τη σοβαρότητα και υπευθυνότητα για αυτό που θέλει να επιτύχει, κάτι που εκτιμάται ότι θα βοηθήσει στην αύξηση του χρόνου ενασχόλησης με το παιχνίδι.

Η τελευταία παράγραφος αποτελεί και το “διασκεδαστικό” μέρος του παιχνιδιού. Εδώ ακριβώς βρίσκεται και η συνεισφορά της εργασίας στον εκπαιδευτικό χώρο. Δεν θα χρησιμοποιηθούν φανταστικοί κόσμοι, ούτε θα γραφτούν φανταστικά σενάρια ώστε να κάνουν “ευκολότερη” την ενασχόληση των εκπαιδευομένων με το παιχνίδι.

Αντιθέτως, θα υπάρξει προσπάθεια ώστε να αποδοθούν, όσο γίνεται πιο ρεαλιστικά, οι απαιτούμενες συνθήκες εκπαίδευσης ώστε να “προσγειωθεί” ο εκπαιδευόμενος και να “μεταφερθεί” στον σύγχρονο κόσμο, από το δωμάτιο του σπιτιού του. Στα επόμενα κεφάλαια θα αναλυθεί η σκέψη και η μελέτη που προηγήθηκε ώστε να επιλεγεί αυτός ο σχεδιασμός.

1.4 Βασική Ορολογία

Παρακάτω παραθέτονται, αλφαβητικά, οι βασικοί τεχνικοί όροι, οι οποίοι θα αναφέρονται συχνά σε όλο το περιεχόμενο της διπλωματικής εργασίας, καθώς και μια περιεκτική περιγραφή τους:

- ✔ **Android NDK:** Το Android NDK (*Native Development Kit*) αφορά ένα σύνολο εργαλείων & μεταγλωττιστών, τα οποία επιτρέπουν την ανάπτυξη εφαρμογών στο Android με τις γλώσσες προγραμματισμού C & C++.

- ✔ **Animation:** Αφορά μια διαδικασία κατά την οποία ένα σύνολο ειδικά επεξεργασμένων εικόνων αναπαράγονται σε μια συσκευή εξόδου – π.χ. μόνιτορ – σε συγκεκριμένα χρονικά διαστήματα, ώστε να δώσουν την εντύπωση κίνησης.

- ✔ **API:** Το API (Application Programming Interface), αλλιώς και Διεπαφή Προγραμματισμού Εφαρμογών, είναι ένα ενδιάμεσο λογισμικό το οποίο παρέχει μια σαφώς καθορισμένη λειτουργικότητα σε άλλες εφαρμογές, χωρίς να απαιτείται ή και να παρέχεται η πρόσβαση στον κώδικα που υλοποιεί την λειτουργικότητα αυτή.

- ✔ **Avatar:** Είναι η γραφική παρουσίαση του κύριου χαρακτήρα ενός παιχνιδιού. Παρέχεται ως επιλογή κυρίως στα RPG παιχνίδια και συνήθως επιλέγεται από τον χρήστη που διαχειρίζεται τον χαρακτήρα.

- ✔ **Background:** Αλλιώς και υπόβαθρο, αποτελεί μια εικόνα 2 διαστάσεων (2D), η οποία χρησιμοποιείται για να καλύψει την επιφάνεια (ή το μεγαλύτερο μέρος της), πάνω από την οποία σχεδιάζονται τα στοιχεία μια εφαρμογής. Συνήθως αφορά εικόνα μεγάλων διαστάσεων ή μικρής εικόνας με επαναλαμβανόμενο μοτίβο.

- ✔ **C++:** Είναι μια υψηλού επιπέδου γλώσσα προγραμματισμού, η οποία σχεδιάστηκε αρχικά ως επέκταση της γλώσσας C, ως μια “C με κλάσεις”, όπως αναφερόταν αρχικά. Διαθέτει αντικειμενοστρεφή και διαδικασιακά χαρακτηριστικά ενώ διατηρεί συμβατότητα με την γλώσσα C. Είναι compiled γλώσσα προγραμματισμού και υπάρχουν διάφοροι compilers, εμπορικοί (MSVC) και δωρεάν (MinGW), οι οποίοι ακολουθούν ένα σύνολο κανόνων (standard) που υπάρχει για κάθε έκδοση της γλώσσας. Η γλώσσα C++ σχεδιάστηκε κυρίως για τη διαχείριση πόρων, μνήμης και μέγιστης απόδοσης σε χαμηλό επίπεδο.

- ✔ **Command line:** Αλλιώς και γραμμή εντολών, αφορά ένα πρόγραμμα το οποίο επιτρέπει την εκτέλεση εντολών σε μορφή κειμένου που παρέχουν οι χρήστες. Το πρόγραμμα *CMD* (αλλιώς και Command Prompt ή απλά γραμμή εντολών) των Windows αφορά ένα χαρακτηριστικό παράδειγμα.

- ✔ **Compiled γλώσσα προγραμματισμού:** Οι γλώσσες προγραμματισμού οι οποίες χρησιμοποιούν compilers. Ένα παράδειγμα είναι η C++.

- ✔ **Compiler:** Ο compiler (μεταγλωττιστής) είναι ένα λογισμικό το οποίο μεταγλωττίζει *πηγαίο κώδικα* μιας γλώσσα προγραμματισμού, σε *κώδικα μηχανής*, το οποίο μπορεί να εκτελεστεί (εκτελέσιμο πρόγραμμα).

- ✔ **Cross-platform:** Ανατρέξτε στον όρο *Ανεξάρτητο πλατφόρμας.*

- ✔ **Device context:** Ανατρέξτε στον όρο *Πλαίσιο συσκευής*.

- ✔ **Driver:** Ανατρέξτε στον όρο *Οδηγός*.

- ✔ **Dynamic library:** Ανατρέξτε στον όρο *Δυναμική βιβλιοθήκη*.

- ✔ **Fog of War:** Είναι η ικανότητα των βιντεοπαιχνιδιών να αποκρύπτουν, με γραφικό τρόπο, περιοχές τις οποίες δεν έχει επισκεφθεί ο χαρακτήρας κάτι που προσθέτει στην ατμόσφαιρα, στον ρεαλισμό και τη στρατηγική.

- ✔ **Full screen:** Ανατρέξτε στον όρο *Πλήρης οθόνη*.

- ✔ **Game Design Document:** Αφορά ένα έντυπο ή ηλεκτρονικό έγγραφο, του οποίου η δημιουργία – από μια ομάδα ανάπτυξης – συνήθως προηγείται της σχεδίασης μιας εφαρμογής. Το έγγραφο αυτό περιγράφει το περιεχόμενο της εφαρμογής, τον τρόπο σχεδίασης του, τις διαδικασίες που θα περιέχει, το σενάριο, ενδεχομένως εικόνες και διαγράμματα και γενικά αφορά έναν *οδηγό* σχεδίασης για τα μέλη της ομάδας που συνεισφέρουν στην ανάπτυξη π.χ. προγραμματιστές, σχεδιαστές κλπ.

- ✔ **Game logic programming:** Ανατρέξτε στον όρο *Προγραμματισμός λογικής παιχνιδιού*.

- ✔ **Gameplay:** Αφορά τον τρόπο με τον οποίο οι παίχτες παίζουν ένα βιντεοπαιχνίδι. Κάθε παιχνίδι έχει σαφώς ορισμένους κανόνες και διαδικασίες με τις οποίες ο παίχτης πρέπει να αλληλεπιδράσει ώστε να ξεπεράσει τα εμπόδια και τις προκλήσεις που του παρουσιάζονται.

- ✔ **GLSL:** Είναι μια γλώσσα προγραμματισμού υψηλού επιπέδου, η οποία επιτρέπει τον προγραμματισμό των καρτών γραφικών, μέσω ενός συντακτικού το οποίο βασίζεται στη γλώσσα προγραμματισμού C. Ενσωματώθηκε το 2004 στην έκδοση 2.0 του OpenGL, με σκοπό την υποστήριξη shaders σε εφαρμογές γραφικών.

- ✔ **GPU:** Αφορά τα αρχικά Graphics Processing Unit . Αναφέρεται κοινώς ως *κάρτα γραφικών*. Είναι ένα ειδικό ολοκληρωμένο κύκλωμα, σχεδιασμένο να διαχειρίζεται ταχύτατα την παραγωγή εικόνων για συσκευές οθόνης. Πλέον, χρησιμοποιούνται σε ένα μεγάλο εύρος συσκευών, από Η/Υ μέχρι κινητά και κονσόλες βιντεοπαιχνιδιών.

- ✔ **GUI:** Είναι τα αρχικά του Graphical User Interface. Αφορά μια μορφή διεπαφής μεταξύ χρήστη και εφαρμογής, η οποία επιτρέπει την αλληλεπίδραση μεταξύ τους – συνήθως μέσω κλικ σε περιοχές γραφικών – επιτρέποντας σε μια εφαρμογή να αντιδρά στις ενέργειες των χρηστών.

- ✔ **Hardware:** Αποτελεί το υλικό μέρος ενός υπολογιστικού συστήματος, για παράδειγμα η μητρική μονάδα, ο επεξεργαστής, η οθόνη κλπ.

- ✔ **IDE:** Αποτελώντας τα αρχικά του Integrated Development Environment, αφορά ένα ολοκληρωμένο περιβάλλον ανάπτυξης το οποίο συνδυάζει συχνές λειτουργίες που απαιτούνται κατά την ανάπτυξη εφαρμογής, όπως είναι η επεξεργασία κώδικα, παραγωγή εκτελέσιμων και αποσφαλμάτωση.

- ✔ **License:** Ανατρέξτε στον όρο *Άδεια χρήσης*.

- ✔ **Mainframe:** Τα mainframes είναι μεγάλα συστήματα υπολογιστών, τα οποία χρησιμοποιούνται για συγκεκριμένες εφαρμογές, για παράδειγμα μαζική επεξεργασία δεδομένων. Έχουν περισσότερη επεξεργαστική ισχύ από τα τυπικά υπολογιστικά συστήματα.

- ✔ **Mockup:** Στην σχεδίαση γραφικών, το mockup ερμηνεύεται ως μια γραφική προσέγγιση των τελικών γραφικών και χρησιμοποιούνται για να αποδώσουν, σε κάποιο βαθμό, το τελικό προϊόν χωρίς όμως να επενδυθεί ιδιαίτερος χρόνος στη σχεδίαση τους.

- ✔ **OpenGL:** Το OpenGL (Open Graphics Library) είναι ένα cross-platform API το οποίο χρησιμοποιείται για σχεδίαση 2D/3D γραφικών. Το κύριο χαρακτηριστικό του είναι η άμεση επικοινωνία με τις κάρτες γραφικών (GPU) ώστε να επιτυγχάνει ταχύτατες επιδόσεις μέσω hardware. Συγκεκριμένα, αφορά ένα specification (σετ κανόνων και εντολών) και κάθε εταιρεία καρτών γραφικών (ή vendor) υλοποιεί την δική της έκδοση OpenGL, σε μορφή *δυναμικής* βιβλιοθήκης, την οποία παρέχει με την μορφή driver. Στην συνέχεια, οι δημιουργοί υλοποιούν τις εφαρμογές γραφικών τους, κάνοντας χρήση του OpenGL API χωρίς να απαιτείται να ξέρουν τι κάρτα γραφικών έχουν ή που θα εκτελεστεί το πρόγραμμα τους.

- ✔ **Portable:** Ανατρέξτε στον όρο *Ανεξάρτητο πλατφόρμας*.

- ✔ **Preprocessor directive:** Ανατρέξτε στον όρο *Εντολές προεπεξεργαστή*.

- ✔ **Role Playing Game:** Αλλιώς και RPG, είναι παιχνίδια με συγκεκριμένα συστήματα κανόνων, στα οποία οι παίχτες παίζουν τον ρόλο κάποιων χαρακτήρων σε κάποιο φανταστικό κόσμο. Οι παίχτες λαμβάνουν αποφάσεις για γεγονότα που παρουσιάζονται μπροστά τους, τα οποία επιδρούν είτε στους ίδιους είτε στον κόσμο γενικότερα. Το κύριο χαρακτηριστικό των παιχνιδιών RPG είναι η παρουσία στατιστικών, μέσω των οποίων λαμβάνονται όλες οι αποφάσεις.

- ✔ **Script:** Πρόκειται για αρχεία *σεναρίων*, τα οποία επιτρέπουν την αυτοματοποιημένη εκτέλεση άλλων προγραμμάτων ή εντολών του λειτουργικού συστήματος. Χρησιμοποιούνται για να απλουστεύσουν διαδικασίες που γίνονται συχνά. Τα αρχεία *batch* (με επέκταση *.bat*) στα Windows, καθώς και τα αρχεία *bash* (με συνήθη επέκταση *.sh*) στο Linux αποτελούν παραδείγματα script αρχείων.

- ✔ **Shader:** Πρόκειται για προγραμματιστικό κώδικα, ο οποίος πρόκειται να εκτελεστεί στον επεξεργαστή της κάρτας γραφικών (GPU). Υπάρχουν διαφορετικοί shaders για κάθε χρήση, για παράδειγμα ο fragment shader εκτελείται για κάθε pixel της εφαρμογής, ενώ ο vertex shader εκτελείται για κάθε vertex (σημείο) των γεωμετρικών αντικειμένων της εφαρμογής.

- ✔ **Sprite:** Πρόκειται για γραφικά δυο (2) διαστάσεων, τα οποία αποτελούν αυτόνομα τμήματα γραφικών μιας οθόνης ενός βιντεοπαιχνιδιού. Συνήθως πρόκειται για κινούμενα ή εν μέρει διαφανή γραφικά, στα οποία αποδίδονται (συνήθως) μηχανισμοί κίνησης και συμπεριφοράς.

- ✔ **Unicode:** Πρόκειται για ένα διεθνές πρότυπο *κωδικοποίησης χαρακτήρων* το οποίο υποστηρίζει τους χαρακτήρες όλων των γλωσσών του κόσμου. Κάθε χαρακτήρας, από τους χιλιάδες σε όλον τον κόσμο, διαθέτει τον δικό του *αριθμό* στην κωδικοποίηση Unicode. Χρησιμοποιείται συχνά στις ιστοσελίδες αλλά και στις εφαρμογές, για την υποστήριξη χαρακτήρων που ανήκουν σε άλλες κωδικοποιήσεις χαρακτήρων που (συνήθως) δεν υπάρχουν εγκατεστημένοι (ως γλώσσες) στο σύστημα του χρήστη.

- ✔ **Vertex:** Αποτελεί την πληροφορία με την οποία περιγράφονται τα γεωμετρικά σχήματα (π.χ. τρίγωνα, κύβοι) σε μια OpenGL εφαρμογή. Ένα γεωμετρικό σχήμα αποτελείται από έναν αριθμό από vertices. Οι θέσεις των γεωμετρικών σχημάτων στον χώρο, το χρώμα τους, τα γραφικά τους και πολλές ακόμη πληροφορίες περιγράφονται από τα vertices τους.

- ✔ **Web based:** Είναι μια εφαρμογή, η οποία εκτελείται σε web browsers. Δεν απαιτείται εγκατάσταση της εφαρμογής στην περίπτωση αυτή, καθώς η εφαρμογή είναι μέρος της ιστοσελίδας που επισκέφθηκε ο χρήστης.

- ✔ **Web browser:** Πρόκειται για εκτελέσιμες εφαρμογές, οι οποίες χρησιμοποιούνται για να κατεβάζουν και να εμφανίζουν ιστοσελίδες στη συσκευή του χρήστη. Μια τέτοια εφαρμογή είναι ο Google Chrome.

- ✔ **XML:** Από τα αρχικά *Extensible Markup Language*, αφορά μια γλώσσα σήμανσης *(markup language),* η οποία χρησιμοποιείται για την δημιουργία οργανωμένου κειμένου, σε μια μορφή που είναι κατανοητή και στους ανθρώπους και στον υπολογιστή και ταυτόχρονα διεθνώς συμβατή. Τα αρχεία στα οποία γράφονται οι εντολές έχουν συνήθως επέκταση *.xml.*

- ✔ **Άδεια χρήσης:** Αλλιώς και software license, περιγράφει τα δικαιώματα τα οποία καθορίζουν οι δημιουργοί του software ως προς τη χρήση και διανομή του προς τρίτους. Ένα τυπικό δικαίωμα χρήσης είναι η χρήση ενός λογισμικού για εμπορικούς σκοπούς.

- ✔ **Ανεξάρτητο πλατφόρμας:** Ένα λογισμικό είναι ανεξάρτητο πλατφόρμας όταν έχει υλοποιηθεί για πολλαπλές πλατφόρμες (λειτουργικά συστήματα ή συσκευές). Επίσης, ένα λογισμικό μπορεί να θεωρηθεί ότι είναι cross-platform εάν απλά ο πηγαίος κώδικας του μπορεί να χρησιμοποιηθεί από τους compilers της ίδιας γλώσσας προγραμματισμού άλλων συστημάτων (λειτουργικών συστημάτων, συσκευών κ.α.). Ένα cross-platform λογισμικό μπορεί να χωριστεί σε δυο βασικούς (2) τύπους:

    1. σε λογισμικό το οποίο απαιτεί ξεχωριστό compilation (παραγωγή εκτελέσιμου κώδικα) για κάθε πλατφόρμα. (π.χ. μέσω C++ compiler)
    2. σε λογισμικό το οποίο θα χρειαστεί μια (1) μόνο φάση compilation και κατόπιν θα μπορεί να εκτελεστεί σε όλες τις πλατφόρμες. (π.χ. μέσω Java JVM)

- ✔ **Αντικειμενοστρεφής προγραμματισμός:** Αλλιώς και object-oriented programming (OOP). Αφορά ένα υπόδειγμα προγραμματισμού το οποίο βασίζεται στην έννοια των “αντικειμένων” (ή objects), τα οποία περιέχουν δεδομένα σε μορφή *ιδιοτήτων* και κώδικα σε μορφή *μεθόδων*. Το κύριο χαρακτηριστικό των αντικειμενοστρεφών γλωσσών προγραμματισμού είναι ότι οι προγραμματιστές καλούν τις μεθόδους των αντικειμένων που φτιάχνουν, ενώ οι μέθοδοι αλληλεπιδρούν με τις ιδιότητες. Δημοφιλείς γλώσσες που υποστηρίζουν αντικειμενοστρεφή προγραμματισμό είναι η C++ και η Java.

- ✔ **Δυναμική βιβλιοθήκη:** Αφορά αρχεία λογισμικού με επέκταση .dll (για τα Windows), τα οποία παρέχουν λειτουργικότητα σε άλλες εφαρμογές. Περισσότερες από μια εφαρμογές μπορούν να χρησιμοποιούν την ίδια δυναμική βιβλιοθήκη, ακόμη και ταυτόχρονα.

- ✔ **Εντολές προεπεξεργαστή:** Αφορά εντολές στον πηγαίο κώδικα της γλώσσας προγραμματισμού, οι οποίες ωστόσο μεταγλωττίζονται πρώτες, πριν αρχίσει η διαδικασία μεταγλώττισης της υπόλοιπης εφαρμογής από τον μεταγλωττιστή. Η κύρια χρήση τους είναι συνήθως να απομονώσουν τμήματα κώδικα πριν αρχίσει η διαδικασία της μεταγλώττισης π.χ. μεταγλώττιση κώδικα για διαφορετικά λειτουργικά συστήματα.

- ✔ **Κάρτα γραφικών:** Ανατρέξτε στον όρο *GPU*.

- ✔ **Μεταγλώττιση:** Αφορά την διαδικασία κατά την οποία δημιουργείται ένα αρχείο εκτελέσιμου κώδικα από αρχεία πηγαίου κώδικα. Την διαδικασία αυτή πραγματοποιούν προγράμματα που ονομάζονται compilers.

- ✔ **Μηχανή παιχνιδιών:** Πρόκειται για ένα σύνολο από αλληλένδετα μέρη λογισμικού, τα οποία χρησιμοποιούνται συνεργατικά για την σχεδίαση και ανάπτυξη βιντεοπαιχνιδιών. Τέτοια μέρη μπορούν να είναι η μηχανή σχεδίασης (renderer), μηχανή φυσικής, σύστημα ήχου, animation, ασύρματη δικτύωση και πολλά άλλα τα οποία περιγράφονται συνήθως ως βασικές λειτουργίες μια μηχανής παιχνιδιών.

- ✔ **Οδηγός:** Αλλιώς και driver, αφορά μια προγραμματιστική διεπαφή (API) με την οποία γίνεται διαχείριση χαμηλού επιπέδου, συνήθως σε κάποιο υλικό (hardware) μέρος μιας συσκευής π.χ. driver της κάρτα γραφικών.

- ✔ **Παιχνίδια σοβαρού σκοπού:** Ένα παιχνίδι σοβαρού σκοπού αφορά ένα παιχνίδι, το οποίο δεν έχει αποκλειστικό σκοπό μόνο την διασκέδαση. Ο όρος “σοβαρό” αναφέρεται στην δευτερεύουσα ιδιότητα των παιχνιδιών αυτών, η οποία είναι αυτή της επίτευξης κάποιου στόχου π.χ. της εκπαίδευσης χρηστών μέσω της ψυχαγωγίας που προσφέρει το παιχνίδι.

- ✔ **Πλαίσιο συσκευής:** Είναι μια προγραμματιστική δομή που συντάσσεται μέσα σε μια εφαρμογή και η οποία περιγράφει τις γραφικές ιδιότητες του παραθύρου στο οποίο θα εκτελεστεί η εφαρμογή.

- ✔ **Πλαίσιο σχεδίασης:** Αποτελεί ένα σαφώς καθορισμένο σύνολο ιδιοτήτων και χαρακτηριστικών, τα οποία θα πρέπει να εφαρμοστούν σε ένα παιχνίδι σοβαρού σκοπού ώστε να υπάρχει μια ουσιαστική καθοδήγηση πριν και κατά τη σχεδίαση του παιχνιδιού, ενώ μπορεί να χρησιμοποιηθεί και στην αξιολόγηση του παιχνιδιού, μετά το πέρας της ανάπτυξης του. Υπάρχουν διάφορα πλαίσια σχεδίασης όπως το EFM, CMX, Scratch, Conceptual framework κ.α.

- ✔ **Πλήρης οθόνη:** Πρόκειται για μια διαδικασία, κατά την οποία μια εφαρμογή, μέσω προγραμματιστικών εντολών, καταλαμβάνει ολόκληρη την διαθέσιμη οθόνη για αποκλειστική της χρήση.

- ✔ **Προγραμματισμός λογικής παιχνιδιού:** Αφορά τους μηχανισμούς μιας εφαρμογής, οι οποίοι υλοποιούν τον τρόπο λειτουργίας & συμπεριφοράς της. Για παράδειγμα, η τεχνητή νοημοσύνη (AI), η μηχανή φυσικής (*physics engine*), η ανίχνευση συγκρούσεων (*collision detection*), η διαχείριση επιπέδων και η συμπεριφορά χαρακτήρων, αποτελούν μέρος του *προγραμματισμού λογικής* ενός παιχνιδιού.

1.5 Διάρθρωση της μελέτης

Στο *κεφάλαιο 2*, θα γίνει μια έρευνα σε παιχνίδια σοβαρού σκοπού, θα παρουσιαστούν κάποια από αυτά και θα ακολουθήσει αξιολόγηση τους με βάση το πλαίσιο σχεδίασης ώστε να διαπιστωθούν πλεονεκτήματα & μειονεκτήματα.

Στο *κεφάλαιο 3*, θα παρουσιαστεί ο σχεδιασμός του παιχνιδιού, θα αναλυθούν οι επιλογές που έγιναν κατά την διαδικασία και θα παρουσιαστεί η γενική ιδέα του παιχνιδιού, σύμφωνα με το πλαίσιο σχεδίασης.

Στο *κεφάλαιο 4*, θα πραγματοποιηθεί η υλοποίηση του παιχνιδιού. Θα παρουσιάζονται και θα αναλύονται τα σημαντικότερα τμήματα του κώδικα της εφαρμογής, καθώς και η λογική πίσω από κάθε ενέργεια.

Στο *κεφάλαιο 5*, θα παρουσιαστεί η υλοποιημένη μορφή του παιχνιδιού, με τη μορφή σχετικών εικόνων, περιγραφικών κειμένων και ενδεχομένως κώδικα.

Στο *κεφάλαιο 6*, θα πραγματοποιηθεί μια αξιολόγηση του παιχνιδιού στο επιλεγμένο πλαίσιο αξιολόγησης, καθώς και αποτελέσματα δοκιμών σε χρήστες.

Στο *κεφάλαιο 7*, παρουσιάζουμε τα τελικά συμπεράσματα από την εργασία μας, τα αποτελέσματα της, τι αντίκτυπο είχε, εάν πέτυχε τον σκοπό της, ανάλυση της αξιολόγησης, τυχόν βελτιώσεις και ποια είναι τα σχέδια για το μέλλον.

2 Παιχνίδια Σοβαρού Σκοπού

2.1 Εισαγωγή

Τα παιχνίδια σοβαρού σκοπού σχεδιάζονται για διαφορετικό σκοπό από αυτόν της απλής διασκέδασης. Αφορούν παιχνίδια διανόησης που παίζονται μέσω υπολογιστή, βασισμένα σε συγκεκριμένους κανόνες και χρησιμοποιούν την διασκέδαση για να εκπαιδεύσουν τους χρήστες τους σε διάφορους τομείς όπως εκπαίδευση, υγεία και άλλους τομείς (Zyda, 2005).

Κάποιοι τομείς στους οποίους έχουν χρησιμοποιηθεί τα παιχνίδια σοβαρού σκοπού είναι οι παρακάτω:

- ✔ **Στρατός**
- ✔ **Εκπαίδευση**
- ✔ **Επιχειρήσεις**
- ✔ **Επιστήμη**
- ✔ **Υγεία**
- ✔ **Διαχείριση καταστάσεων**
- ✔ **Πολεοδομία**
- ✔ **Μηχανική**
- ✔ **Πολιτική**
- ✔ **Αλλαγή συμπεριφορών**

Τα βιντεοπαιχνίδια που έχουν ως κύριο σκοπό τη διασκέδαση αποτελούνται μόνο από σενάριο, πολυμεσικό υλικό (γραφικά, ήχος, βίντεο) και λογισμικό. Η ομάδα ανάπτυξης συγκεντρώνει όλα αυτά τα στοιχεία σε ένα τελικό προϊόν, το ίδιο το παιχνίδι (Zyda, 2005).

Η ανάπτυξη των παιχνιδιών σοβαρού σκοπού ακολουθούν την ίδια διαδικασία για να σχεδιαστούν και να υλοποιηθούν, ωστόσο περιλαμβάνουν και το στοιχείο παιδαγωγίας – διαδικασίες οι οποίες εκπαιδεύουν τους χρήστες, μεταδίδοντας συγκεκριμένη γνώση ή ικανότητες σε αυτούς κατά τη χρήση του παιχνιδιού. Αυτή ακριβώς η προσθήκη μετατρέπει τα παιχνίδια αυτά σε παιχνίδια σοβαρού σκοπού.

2.2 Ιστορική αναδρομή

Την πρώτη αναφορά στον όρο “παιχνίδια σοβαρού σκοπού” την βρίσκουμε αρκετά χρόνια στο παρελθόν, συγκεκριμένα το 1970, στο βιβλίο *Serious Games* του Αμερικανού ερευνητή Clark Abt.

Στο συγκεκριμένο βιβλίο, ο ερευνητής εξερευνά τους τρόπους με τους οποίους τα παιχνίδια θα μπορούσαν να εκπαιδεύουν τους χρήστες. Συγκεκριμένα, ο Clark Abt έγραψε τον παρακάτω, ίσως πρώτο, ορισμό των παιχνιδιών σοβαρού σκοπού (1970):

“τα παιχνίδια μπορούν να παιχτούν σοβαρά ή χωρίς λόγο. Ανησυχούμε για τα παιχνίδια σοβαρού σκοπού υπό την έννοια ότι τα παιχνίδια αυτά έχουν έναν ρητό και προσεκτικά σχεδιασμένο εκπαιδευτικό σκοπό και δεν ενδείκνυται να παιχτούν μόνο για διασκέδαση. Αυτό δεν σημαίνει ότι τα παιχνίδια σοβαρού σκοπού δεν είναι, ή δεν πρέπει να είναι, διασκεδαστικά”.

Λίγα χρόνια αργότερα, έκανε την εμφάνιση του το βιβλίο *The New Alexandria Simulation: A Serious Game of State and Local Politics* (Donald R. Jansiewics, 1973). Στο βιβλίο αυτό αναλύεται ο τρόπος κατά τον οποίο θα μπορούσε να παιχτεί ένα παιχνίδι που σχεδιάστηκε για την εκπαίδευση των πολιτικών μηχανισμών των ΗΠΑ. Αν και δεν δημιουργήθηκε ποτέ παιχνίδι σοβαρού σκοπού βασισμένο στις θεωρίες του βιβλίου, το συγκεκριμένο βιβλίο χρησιμοποιείται ακόμη (αναθεωρημένο).

Τα πρώτα παιχνίδια που είχαν εκπαιδευτικό σκοπό, όχι όμως και χαρακτηριστικά διασκέδασης, έκαναν την εμφάνιση τους στα χρόνια του Ψυχρού Πολέμου, ήταν δηλαδή προορισμένα για στρατιωτικούς σκοπούς και συνήθως είχαν χαρακτήρα προσομοίωσης.

Ένα τέτοιο παιχνίδι ήταν και το *HUTSPIEL* (1955), το οποίο ήταν ένα παιχνίδι στρατηγικής μεταξύ δυο (2) παιχτών, του red και του blue, που επέτρεπε τον πειραματισμό με χρήση πυρηνικών όπλων σε παγκόσμια κλίμακα. Κάθε παίχτης αντιπροσώπευε βασικά είτε το NATO είτε την USSR.

Μερικά χρόνια αργότερα, ο Clark Abt (ο οποίος αναφέρθηκε παραπάνω) σχεδίασε το παιχνίδι για υπολογιστές *T.E.M.P.E.R.* (1961), το οποίο ήταν από τα πρώτα παιχνίδια εκπαιδευτικού σκοπού (όχι ακριβώς σοβαρού σκοπού), το

οποίο χρησιμοποιήθηκε για στρατιωτικούς σκοπούς, συγκεκριμένα για την μελέτη του Ψυχρού Πολέμου σε συγκρούσεις παγκόσμιας κλίμακας.

Παρόμοια παιχνίδια αναπτύχθηκαν την εποχή εκείνη, η συντριπτική πλειοψηφία των οποίων ήταν προορισμένη για στρατιωτική χρήση και δεν ήταν ανοιχτά στο ευρύ κοινό. Για παράδειγμα το *AGILE-COIN* (1965), το οποίο έκανε προσομοίωση εξεγέρσεων σε υποθετικά κράτη. Τα παιχνίδια αυτά, αν και δεν ήταν σοβαρού σκοπού με την ευρεία έννοια που υπάρχει σήμερα, μπορούν να θεωρηθούν ως οι πρόγονοι των εμπορικών παιχνιδιών προσομοίωσης που εμφανίστηκαν αρκετά χρόνια αργότερα.

2.3 Τα πρώτα παιχνίδια σοβαρού σκοπού

Ένα από τα πρώτα παιχνίδια (Εικόνα 1) που είχαν χαρακτήρα σοβαρού σκοπού ήταν το παιχνίδι για υπολογιστές *The Oregon Trail* (Rawitsch, D., Heinemann, B.,& Dillenberger P., 1971) και εκδόθηκε από την MECC (Minnesota Educational Computing Consortium), έναν Αμερικάνικο οργανισμό ο οποίος παρείχε υπηρεσίες πληροφορικής σε σχολεία της πολιτείας της Μινεσότα, ΗΠΑ.

```
MØNDAY MAY 10 1847

TØTAL MILEAGE IS 575
FØØD          BULLETS        CLØTHING       MISC. SUPP.     CASH
 46            1090            40              45             205
DØ YØU WANT TØ (1) STØP AT THE NEXT FØRT, (2) HUNT, ØR (3) CØNTINUE
?2
TYPE BANG  BANG
RIGHT BETWEEN THE EYES---YØU GØT A BIG ØNE!!!!
WATCH YØUR CALØRIES TØNIGHT!!!
DØ YØU WANT TØ EAT (1) PØØRLY  (2) MØDERATELY
ØR (3) WELL?2
HAIL STØRM---SUPPLIES DAMAGED
```

Εικόνα 1: The Oregon Trail (1971)

Wikipedia, 2018, https://en.wikipedia.org/wiki/File:TheOregonTrail1971Gameplay.png

Το παιχνίδι δίδασκε την Αμερικάνικη ιστορία του 19ου αιώνα σε μαθητές που είχαν πρόσβαση στο mainframe που υπήρχε εγκατεστημένο το παιχνίδι. Η πρώτη έκδοση ήταν σε μορφή κειμένου, όχι για οθόνες υπολογιστών αλλά για ειδικές συσκευές που ονομάζονταν teleprinters, οι οποίες χρησιμοποιούνταν για να στέλνουν και να λαμβάνουν μηνύματα μέσω υπολογιστή σε έντυπη μορφή.

Ακολούθησαν και άλλες εκδόσεις, με αυτήν του 1974 να προσθέτει περισσότερα στοιχεία ρεαλισμού (αληθινά γεγονότα, καιρικές συνθήκες) ενώ προστέθηκε και οδηγός για τους εκπαιδευτές (!). Το 1985 αναπτύχθηκε μια έκδοση για τον Apple II όπου πλέον υπήρχαν γραφικά πλήρους οθόνης και η οποία έκδοση πούλησε δεκάδες εκατομμύρια αντίτυπα.

Ένα παιχνίδι σοβαρού σκοπού για την υγεία ήταν το *Captain Novolin* (Raya Systems, 1992, SNES), το οποίο σχεδιάστηκε για τα άτομα που πάσχουν από διαβήτη και τρόπους διαχείρισης του (Εικόνα 2). Ο κεντρικός ήρωας του παιχνιδιού ήταν ένας υπεράνθρωπος, ο οποίος όμως έπασχε από διαβήτη. Σκοπός των παιχτών ήταν να αποφεύγουν την κατανάλωση ανεπιθύμητων τροφών και να φροντίζουν για τη σωστή και έγκαιρη δόση ινσουλίνης, με βάση τις τροφές που καταναλώνει ο ήρωας τους. Το παιχνίδι προοριζόταν για παιδιά με διαβήτη, ηλικίας 8 με 14 και ήταν αρκετά εύκολο στη χρήση του.

Εικόνα 2: Captain Novolin (1992)

Wikipedia, 2016, https://en.wikipedia.org/w/index.php?curid=48360713

Το 2002 έκανε την εμφάνιση του το πρώτο παιχνίδι σοβαρού σκοπού που ήταν τεχνολογικά εξελιγμένο σε πολλά επίπεδα, το *America's Army* (Εικόνα 3). Χρησιμοποιώντας την πρώτη έκδοση της δημοφιλούς (σήμερα) μηχανής παιχνιδιών Unreal (1998), το America's Army κατάφερε να αναπαράγει εξαιρετικά ρεαλιστικά 3D γραφικά και ηχητικά εφέ.

Το παιχνίδι έδινε την ευκαιρία στους χρήστες να πάρουν μέρος στη βασική στρατιωτική εκπαίδευση. Στην συνέχεια, ο χρήστης μπορούσε να επιλέξει

είτε προχωρημένη εκπαίδευση στις Ειδικές Δυνάμεις ή να παίξει διαδικτυακά, ως στρατιώτης, και να πάρει μέρος σε αποστολές, ως (διαδικτυακό) μέλος μιας ομάδας. Θα μπορούσε να ειπωθεί ότι ήταν ο πρόγονος δημοφιλών πολεμικών προσομοιωτών της σημερινής εποχής όπως π.χ. των πολεμικών παιχνιδιών για υπολογιστές *Call of Duty* και *Medal of Honor*. Η πρώτη του έκδοση ωστόσο, δεν κατάφερε τελικά να παρουσιάσει με τον καλύτερο δυνατό τρόπο όσα υποσχόταν, με αρκετές κριτικές να αφορούν τον ελλειπή ρεαλισμό σε πραγματικές συνθήκες.

Από τότε, έχουν αναπτυχθεί δεκάδες εκδόσεις για το America's Army, με την τελευταία, *America's Army: Proving Grounds* (2015), να διατίθεται δωρεάν, έχοντας οικονομική χορήγηση από την ίδια την κυβέρνηση των ΗΠΑ ενώ χρησιμοποιεί ακόμη την μηχανή Unreal, συγκεκριμένα την 3η έκδοση του 2004.

Η ανάπτυξη, καθώς και η ευρεία διάδοση του παιχνιδιού America's Army ξεκίνησε μια επανάσταση στον τρόπο σκέψης σχετικά με τη χρήση των βιντεοπαιχνιδιών σε μη-διασκεδαστικά πεδία. Επίσης πυροδότησε συζητήσεις σχετικά με την ανάπτυξη της παρούσας τεχνολογίας του παιχνιδιού για την ανάπτυξη μελλοντικών παιχνιδιών σοβαρού σκοπού, καθώς και την προσθήκη του παράγοντα διασκέδασης (Zyda, 2005).

Εικόνα 3: America's Army (2002)
Serious Game Classification: America's Army,
http://serious.gameclassification.com/EN/games/758-Americas-Army/index.html

2.4 Σχεδίαση παιχνιδιού

Η δημιουργία ενός παιχνιδιού είναι ιδιαίτερα περίπλοκη διαδικασία. Το τελικό προϊόν, το παιχνίδι, αποτελείται από πολλά τμήματα τα οποία πρέπει να συνδέονται αρμονικά μεταξύ τους ώστε το τελικό αποτέλεσμα να είναι ελκυστικό, διασκεδαστικό και ταυτόχρονα να παρουσιάζει προκλήσεις στους χρήστες. Τα τμήματα αυτά είναι:

- ✔ **Σενάριο:** Η ιστορία, το περιεχόμενο και οι προκλήσεις του παιχνιδιού.
- ✔ **Πόροι:** Τα γραφικά, ο ήχος, βίντεο, 3D μοντέλα, χάρτες κλπ.
- ✔ **Λογισμικό:** Ο κώδικας του παιχνιδιού.

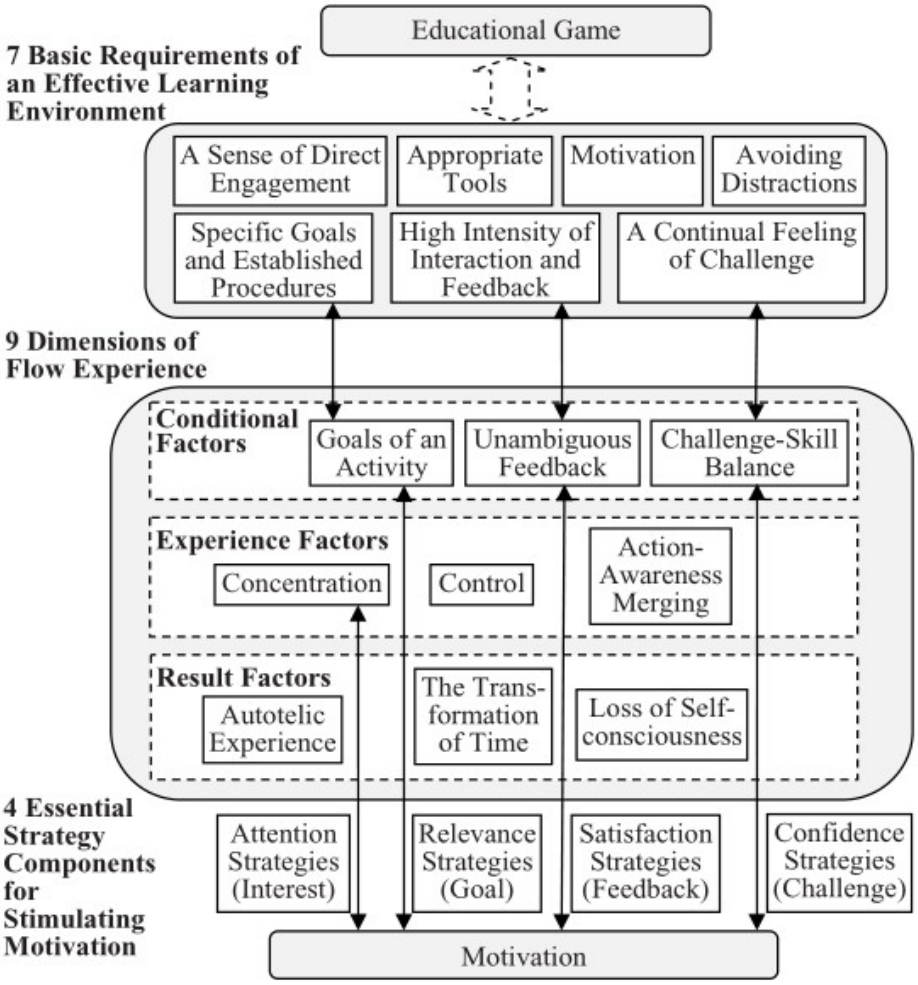
Ιδιαίτερα σε ένα παιχνίδι σοβαρού σκοπού, όπου εισάγεται και το στοιχείο της παιδαγωγίας στο τελικό προϊόν, θα πρέπει να απαντηθούν ερωτήματα όπως το είδος και ο τρόπος που θα ενσωματωθεί ο παράγοντας της διασκέδασης στο εκπαιδευτικό μέρος του παιχνιδιού ή πως θα αναδειχθούν οι μηχανισμοί που θα παρέχουν συνεχές κίνητρο στους παίχτες. Όλα αυτά τα ερωτήματα μπορούν να απαντηθούν εάν επιλεγεί κάποιο *πλαίσιο σχεδίασης* (design framework), το οποίο θα οργανώσει σωστά την ανάπτυξη του παιχνιδιού.

2.5 Το πλαίσιο σχεδίασης EFM

Η εφαρμογή θα αναπτυχθεί με βάση το πλαίσιο σχεδίασης *EFM* (Song, M., & Zhang, S., 2008). Το EFM είναι ακρωνύμιο των παρακάτω:

- ✔ **[E] ffective learning environment**
- ✔ **[F] low experience**
- ✔ **[M] otivation**

Τα παραπάνω αποτελούν τους κύριους άξονες της σχεδίασης ενός παιχνιδιού σοβαρού σκοπού, το οποίο θα βασίζεται στο πλαίσιο σχεδίασης EFM (Εικόνα 4). Κάθε ένας από αυτούς τους άξονες, περιλαμβάνει ένα σύνολο από ιδιότητες και χαρακτηριστικά, τα οποία πρέπει να ακολουθηθούν κατά τον σχεδιασμό του παιχνιδιού και τα οποία αναλύονται παρακάτω.



Εικόνα 4: EFM - Model for Educational Game

EFM: A Model for Educational Game Design, 2008, p.5

Στα παιχνίδια σοβαρού σκοπού που θα παρουσιαστούν, θα αξιοποιηθεί το πλαίσιο σχεδίασης EFM ώστε να αξιολογηθεί ο βαθμός επίτευξης των χαρακτηριστικών του πλαισίου σχεδίασης για κάθε παιχνίδι και να εξαχθούν χρήσιμα συμπεράσματα.

Με βάση τα συμπεράσματα αυτά, θα υπάρχει μια καθαρή εικόνα των δυνατών και αδύναμων σημείων των παιχνιδιών αυτών, τόσο σε επίπεδο διασκέδασης όσο και εκπαίδευσης, κάτι που θα βοηθήσει στη σχεδίαση της προς ανάπτυξη εφαρμογής σοβαρού σκοπού στο πλαίσιο σχεδίασης EFM.

2.5.1 Αποτελεσματικό μαθησιακό περιβάλλον

Ο άξονας αυτός συντελεί στην αποτελεσματικότητα του μαθησιακού περιβάλλοντος (effective learning environment), στο οποίο ενσωματώνονται όλες οι καταστάσεις οι οποίες προάγουν την μάθηση των εκπαιδευομένων. Στο πλαίσιο αυτό περιγράφονται εφτά (7) βασικά χαρακτηριστικά για ένα αποτελεσματικό μαθησιακό περιβάλλον, τα οποία είναι:

- ✔ **Μεγάλος βαθμός αλληλεπίδρασης και ανάδρασης:** Ο εκπαιδευόμενος θα πρέπει να αλληλεπιδράει σε μεγάλο βαθμό με την εφαρμογή, αλλά και να λαμβάνει κατάλληλη ανάδραση (feedback).

- ✔ **Συγκεκριμένοι στόχοι και διαδικασίες:** Θα πρέπει να υπάρχουν σαφώς καθορισμένοι εκπαιδευτικοί στόχοι και διαδικασίες.

- ✔ **Κίνητρο:** Η εφαρμογή θα πρέπει να δίνει κίνητρο στον εκπαιδευόμενο να συνεχίσει την ενασχόληση του με το παιχνίδι.

- ✔ **Συνεχής αίσθηση της πρόκλησης:** Η οποία δεν πρέπει να είναι τόσο δύσκολη ώστε να προκαλεί αίσθημα απελπισίας, ούτε τόσο εύκολη ώστε να προκαλεί πλήξη.

- ✔ **Αίσθημα άμεσης επαφής και αλληλεπίδρασης με το περιβάλλον:** Ο χρήστης θα πρέπει να αισθάνεται ότι αλληλεπιδρά άμεσα με το μαθησιακό περιβάλλον του, να αισθάνεται ότι είναι μέρος του.

- ✔ **Κατάλληλα και σχετικά εργαλεία:** Τα οποία εργαλεία βοηθούν τον εκπαιδευόμενο, χωρίς να τον αποπροσανατολίζουν.

- ✔ **Αποφυγή απόσπασης της προσοχής του εκπαιδευομένου:** Ώστε να μην υπάρχει παρέμβαση στην εκπαιδευτική διαδικασία από παράγοντες που είτε δεν τον διασκεδάζουν είτε δεν τον εκπαιδεύουν.

2.5.2 Εμπειρία ροής

Η θεωρία της ροής (Csikszentmihalyi, 1975) αναπτύχθηκε ως μια μέθοδος κατανόησης και υλοποίησης κινήτρων στη σχεδίαση της εκπαιδευτικής διαδικασίας. Χρησιμοποιήθηκε κυρίως στην ψυχολογία και συμπεριφορά μεταξύ ανθρώπου και μηχανής.

Σύμφωνα με τη θεωρία αυτή, υπάρχουν κάποιες καταστάσεις ροής (flow states), οι οποίες αφορούν ουσιαστικά τις εσωτερικές καταστάσεις στις οποίες βρίσκεται ένας άνθρωπος ο οποίος εκτελεί μια δραστηριότητα. Όταν ένας άνθρωπος βρίσκεται στις καταστάσεις αυτές, τότε ο άνθρωπος αυτός 1) είναι πλήρως αφοσιωμένος στη διαδικασία, και 2) απολαμβάνει πλήρως την διαδικασία. Ο Csikszentmihalyi (1991) οριοθέτησε αυτές τις καταστάσεις ροής σε εννέα (9) διαστάσεις:

- ✔ **Στόχοι της δραστηριότητας:** Κάθε δραστηριότητα θα πρέπει να έχει σαφείς στόχους.

- ✔ **Αναμφισβήτητη ανάδραση:** Οι πληροφορίες που λαμβάνει ένας εκπαιδευόμενος ως αποτέλεσμα της εκτέλεσης μιας δραστηριότητας, να μην επιδέχονται αμφισβήτησης.

- ✔ **Ισορροπία πρόκλησης και δεξιοτήτων:** Όταν οι προκλήσεις έχουν διαβαθμισμένο επίπεδο δυσκολίας το οποίο ταιριάζει στις δεξιότητες του εκπαιδευομένου.

- ✔ **Συγχώνευση δράσης και συνειδητοποίησης:** Όταν ο εκπαιδευόμενος είναι πλήρως απορροφημένος με την τρέχουσα διαδικασία, αλλά ταυτόχρονα είναι πλήρως συνειδητοποιημένος με τις επιλογές του.

- ✔ **Συγκέντρωση:** Όταν ο εκπαιδευόμενος έχει υψηλό βαθμό συγκέντρωσης στην τρέχουσα διαδικασία, μειώνονται δραστικά οι πιθανότητες του να χάσει την προσοχή του σε κάτι άλλο, άσχετο με την τρέχουσα διαδικασία.

- ✔ **Έλεγχος:** Όταν ο εκπαιδευόμενος νιώθει ότι έχει μεγάλο έλεγχο κατά την εκπαιδευτική διαδικασία.

- ✔ **Απώλεια της ατομικής συνείδησης:** Όταν ο εκπαιδευόμενος είναι τόσο αφοσιωμένος στην τρέχουσα διαδικασία ώστε να μην ενεργεί εγωιστικά, αμυντικά ή με αρνητική προδιάθεση.

- ✔ **Η μεταμόρφωση του χρόνου:** Όταν ο εκπαιδευόμενος χάνει την αίσθηση του χρόνου κατά την τρέχουσα διαδικασία. Στην περίπτωση αυτή, ο εκπαιδευόμενος αισθάνεται ότι ο χρόνος επιβραδύνεται ή επιταχύνεται ανάλογα με τη διαδικασία.

- ✔ **Αυτόνομη εμπειρία:** Όταν ο εκπαιδευόμενος αισθάνεται ικανοποίηση κατά την εκτέλεση της διαδικασίας, η οποία λειτουργεί αυτόνομα και ο κύριος σκοπός της είναι απλά η εμπειρία της εκτέλεσης της.

2.5.3 Κίνητρο

Το κίνητρο (motivation) έχει άμεση σχέση με τον λόγο για τον οποίο οι εκπαιδευόμενοι θέλουν να μάθουν ή να εκπαιδευτούν σε ένα αντικείμενο. Κατά τον σχεδιασμό ενός εκπαιδευτικού παιχνιδιού, ένα βασικό ερώτημα είναι πως θα ενσωματωθεί το αίσθημα του κινήτρου αλλά και πως αυτό θα διατηρηθεί, κατά την εκπαιδευτική διαδικασία. Το πλαίσιο σχεδίασης EFM χρησιμοποιεί το μοντέλο κινητήριου σχεδιασμού *A.R.C.S.* (Keller, 2014). Το μοντέλο A.R.C.S. προσδιορίζεται από τέσσερα (4) στρατηγικά συστατικά, τα αρχικά των οποίων αποτελούν το ακρωνύμιο του μοντέλου. Τα συστατικά αυτά είναι τα παρακάτω:

- ✔ **[A] ttention:** Αφορά το ενδιαφέρον του εκπαιδευόμενου. Είναι κρίσιμο η εκπαιδευτική εφαρμογή να κινήσει το ενδιαφέρον και την προσοχή του εκπαιδευόμενου. Με άλλα λόγια, το κίνητρο για τον εκπαιδευόμενο αυξάνεται όταν η εκπαιδευτική εφαρμογή προκαλεί ενδιαφέρον ή περιέργεια για το περιεχόμενο της.

- ✔ **[R] elevance:** Το αποτέλεσμα της εκπαιδευτικής διαδικασίας πρέπει να είναι χρήσιμο ώστε να είναι σχετικό, τόσο με τις απαιτήσεις των εκπαιδευομένων αλλά και με τις απαιτήσεις του πραγματικού κόσμου.

- ✔ **[C] onfidence:** Το συστατικό αυτό βοηθάει τους εκπαιδευόμενους να αναπτύξουν θετικές προσδοκίες για ένα επιτυχές αποτέλεσμα. Αυτό θα τους επιτρέψει να έχουν μεγαλύτερο έλεγχο στη διαδικασία εκπαίδευσης.

- ✔ **[S] atisfaction:** Οι εκπαιδευόμενοι πρέπει να νιώθουν ικανοποίηση όταν επιτυγχάνουν κάτι κατά την εκπαιδευτική διαδικασία. Η σχέση κινήτρου και ικανοποίησης είναι στενά συνδεδεμένη.

2.6 Αξιολόγηση παιχνιδιών σοβαρού σκοπού

Η ανάπτυξη του παιχνιδιού σοβαρού σκοπού θα βασιστεί στο πλαίσιο σχεδίασης EFM, ωστόσο είναι εξαιρετικά χρήσιμο να πραγματοποιηθεί μια επισκόπηση σε υπάρχοντα παιχνίδια σοβαρού σκοπού που κυκλοφορούν στο διαδίκτυο και να βγουν χρήσιμα συμπεράσματα, αξιολογώντας κάποια από αυτά με το επιλεγμένο πλαίσιο σχεδίασης.

Αυτό θα βοηθήσει ώστε να εντοπιστούν πλεονεκτήματα και μειονεκτήματα του κάθε παιχνιδιού, κάτι που θα αξιοποιηθεί στη συνέχεια κατά την ανάπτυξη της εφαρμογής. Ωστόσο, πριν γίνει αυτό θα πρέπει να συγκεντρωθούν οι ιδιότητες που χαρακτηρίζουν το πλαίσιο σχεδίασης EFM ώστε η ανάπτυξη και η αξιολόγηση του παιχνιδιού, αλλά και η αξιολόγηση των παρακάτω παιχνιδιών σοβαρού σκοπού, να γίνει σωστά.

Όπως αναφέρθηκε και παραπάνω, το πλαίσιο σχεδίασης EFM βασίζεται στους παρακάτω τρεις (3) άξονες, καθώς και τις ιδιότητες τους (επιγραμματικά):

Αποτελεσματικό μαθησιακό περιβάλλον

1. Μεγάλος βαθμός αλληλεπίδρασης και ανάδρασης.
2. Συγκεκριμένοι στόχοι και διαδικασίες.
3. Κίνητρο.
4. Συνεχής αίσθηση της πρόκλησης.
5. Αίσθημα άμεσης επαφής και αλληλεπίδρασης με το περιβάλλον.
6. Κατάλληλα και σχετικά εργαλεία.
7. Αποφυγή απόσπασης της προσοχής του εκπαιδευομένου.

Εμπειρία ροής

1. Στόχοι της δραστηριότητας.
2. Αναμφισβήτητη ανάδραση.
3. Ισορροπία πρόκλησης και δεξιοτήτων.
4. Συγχώνευση δράσης και συνειδητοποίησης.
5. Συγκέντρωση.
6. Έλεγχος.
7. Απώλεια της ατομικής συνείδησης.
8. Η μεταμόρφωση του χρόνου.
9. Αυτόνομη εμπειρία.

Κίνητρο

1. Ενδιαφέρον.
2. Συνάφεια.
3. Αυτοπεποίθηση.
4. Ικανοποίηση.

Τα παραπάνω χαρακτηριστικά & ιδιότητες του πλαισίου σχεδίασης EFM θα αξιολογηθούν σε επιλεγμένα παιχνίδια σοβαρού σκοπού ώστε να καταγραφούν τα αδύνατα αλλά και δυνατά σημεία τους.

Κάθε αναφορά σε παιχνίδι, θα ξεκινάει με μια χαρακτηριστική εικόνα μέσα από το παιχνίδι, έπειτα θα ακολουθεί μια αναλυτική περιγραφή του και στη συνέχεια θα αξιολογείται σε όλες τις ιδιότητες & χαρακτηριστικά του πλαισίου σχεδίασης EFM.

Τέλος, θα παρουσιάζονται τα αποτελέσματα της αξιολόγησης σε έναν πίνακα, με χρήση συστήματος likert εξαβάθμιας διαβάθμισης. Τα αποτελέσματα που παρουσίασαν ενδιαφέρον, είτε αρνητικό είτε θετικό, θα είναι και αυτά που θα ληφθούν υπόψιν.

2.7 CodeCombat

2.7.1 Περιγραφή

Το *CodeCombat* (http://codecombat.com) είναι ένα παιχνίδι σοβαρού σκοπού με στοιχεία Role Playing Game (RPG). Αφορά ένα web-based παιχνίδι σοβαρού σκοπού, το οποίο εκπαιδεύει τους χρήστες στις βασικές έννοιες προγραμματισμού σε τρεις (3) δημοφιλείς γλώσσες προγραμματισμού, την Python, την JavaScript και την CoffeeScript και στοχεύει σε ηλικίες από 9+ ετών. Υποστηρίζει πολλές γλώσσες, καθώς και την Ελληνική.

Το παιχνίδι βάζει από πολύ νωρίς τον παίχτη στο κλίμα των RPG παιχνιδιών, οδηγώντας τον στην επιλογή του χαρακτήρα του από μια σειρά διαθέσιμων avatars, στην επιλογή του εξοπλισμού του ενώ τον ενημερώνει για τις *ικανότητες* που διαθέτει ήδη και παράλληλα επιλέγει και τη γλώσσα προγραμματισμού, στην οποία θέλει να εκπαιδευτεί.

Οι ικανότητες του χαρακτήρα εδώ παίζουν τον ρόλο των *μεθόδων* των κλάσεων στις γλώσσες προγραμματισμού, δηλαδή τις δυνατότητες που έχει ήδη και μπορεί να χρησιμοποιεί στις προκλήσεις που θα του παρουσιάζονται.

Το παιχνίδι ξεκινά, μεταφέροντας τον παίχτη στην πρώτη περιοχή, το οποίο είναι ένα σκοτεινό μπουντρούμι (Εικόνα 5), εμφανίζοντας παράλληλα και άλλες (προχωρημένες) περιοχές, οι οποίες όμως παραμένουν αρχικά κλειδωμένες. Η κάθε περιοχή διαθέτει μια σειρά από *επίπεδα,* τα οποία πρακτικά περιέχουν μια ή παραπάνω προκλήσεις, τις οποίες ο χαρακτήρας πρέπει να περάσει, χρησιμοποιώντας τις *ικανότητες* του (όπως ακριβώς σε ένα πραγματικό παίχνιδι RPG). Ωστόσο, εδώ οι ικανότητες είναι μέθοδοι μιας κλάσης και τις οποίες ο εκπαιδευόμενος πρέπει να γράψει σε ένα ειδικό πλαίσιο ώστε να καθοδηγεί τον χαρακτήρα του ανάμεσα στα επίπεδα.

Εικόνα 5: CodeCombat - Πρώτη περιοχή

2.7.2 Αποτελεσματικό μαθησιακό περιβάλλον

Ο *βαθμός αλληλεπίδρασης και ανάδρασης* είναι καλός. Σε κάθε επίπεδο, ο παίχτης μπορεί να κινεί τον χαρακτήρα του όπως θέλει, σύμφωνα με τους στόχους που του παρουσιάζονται, εισάγοντας κώδικα σε ειδικό πλαίσιο κειμένου, ενώ τα αποτελέσματα (feedback) γίνονται άμεσα ορατά. Ο κώδικας που παράγεται εστιάζει περισσότερο στο υπάρχον API του παιχνιδιού και στη χρήση έτοιμων μεθόδων παρά στην υλοποίηση από την αρχή σε βασικά κομμάτια της γλώσσας π.χ. μεταβλητές και έτσι η ανάδραση δεν είναι ιδιαίτερα ουσιαστική.

Υπάρχουν *συγκεκριμένοι στόχοι και διαδικασίες* σε κάθε επίπεδο, οι οποίοι καθορίζουν το είδος των *ικανοτήτων* (ή μεθόδων κλάσης) που μπορεί να χρησιμοποιήσει ο χαρακτήρας μέσω κώδικα. Οι στόχοι αυτοί είναι διαθέσιμοι ανά πάσα στιγμή και έτσι ο χρήστης έχει ξεκάθαρη εικόνα τι πρέπει να κάνει. Κάθε επίπεδο αναφέρει ξεκάθαρα τα προγραμματιστικά θέματα τα οποία θα παρουσιαστούν (π.χ. μεταβλητές, σύνταξη κλπ.)

Η αίσθηση του *κινήτρου* δεν προάγεται ιδιαίτερα και αυτό ξεκινάει από το μηδαμινό σενάριο. Ο χρήστης απλά μετακινείται σε στατικά επίπεδα, λύνοντας προκλήσεις μέσω κώδικα μέχρι να μετακινηθεί σε άλλη, δυσκολότερη (προγραμματιστικά) περιοχή. Δεν υπάρχει fog of war στους χάρτες, τόσο των

επιπέδων όσο και των περιοχών, κάτι που αφαιρεί πολύ από το μυστήριο των RPG παιχνιδιών και εν τέλει του κινήτρου.

Η *αίσθηση της πρόκλησης* δεν είναι ισορροπημένη. Από πολύ νωρίς, παρουσιάζονται πολλές πληροφορίες στον χρήστη, τόσο στο GUI όσο και σε επίπεδο στόχων, κάτι που τον μπερδεύει, δυσκολεύοντας τις προκλήσεις.

Το *αίσθημα άμεσης επαφής και αλληλεπίδρασης με το περιβάλλον* είναι καλό. Ο χρήστης μπορεί άμεσα να δει το σύνολο των *ικανοτήτων* του (ή μεθόδων κλάσεων) που απαιτούνται για να ολοκληρώσει το επίπεδο, ενώ η σελίδα που εξηγεί τους στόχους καθώς και “hints” για το παρόν επίπεδο είναι στη διάθεση του χρήστη και εύκολο να επιλεγούν.

Τα *εργαλεία* που έχουν επιλεγεί έχουν αρνητικά και θετικά. Τα χρώματα είναι λιτά αλλά καλαίσθητα και δεν αποπροσανατολίζουν τον χρήστη, το GUI είναι πρακτικό και ελαφρύ, η μουσική δένει το γραφικό αποτέλεσμα, χωρίς να παρεμβαίνει αρνητικά ενώ η υποστήριξη πολλών γλωσσών διευρύνει τη βάση χρηστών. Ωστόσο, αν και πρόκειται για web based εφαρμογή, δεν είναι φιλική προς τις κινητές συσκευές. Η χρήση του κατά την εκπαίδευση είναι απλή, ωστόσο κάποιες διαδικασίες περισσότερο αποπροσανατολίζουν (π.χ. χρήση RPG αντικείμενων) παρά ενισχύουν είτε την εκπαίδευση είτε την διασκέδαση. Τέλος, η οθόνη δράσης είναι πολύ μικρή και βασίζεται περισσότερο στη μεγέθυνση του παραθύρου ή στα +/- κουμπιά που υπάρχουν για τοπικό zoom.

Η *απόσπαση της προσοχής του εκπαιδευόμενου* δεν συμβαίνει συχνά καθώς τα επίπεδα αφορούν στατικές οθόνες με κινούμενα sprites, ωστόσο η μικρή οθόνη δράσης, σε συνδυασμό με τα πολλά (συνήθως) sprites μπερδεύει τον χρήστη. Επίσης, το παράθυρο των στόχων εμφανίζεται συχνά επάνω στο (μικρό) παράθυρο δράσης, αποπροσανατολίζοντας περαιτέρω τον χρήστη.

2.7.3 Εμπειρία ροής

Οι *στόχοι δραστηριότητας* είναι ξεκάθαροι, τόσο σε εκπαιδευτικό όσο και λειτουργικό επίπεδο. Κάθε επίπεδο αναφέρει ρητά τους εκπαιδευτικούς στόχους, πριν ο χρήστης το επιλέξει, ενώ μέσα στο επίπεδο εμφανίζεται συνέχεια ένα παράθυρο με τους στόχους και πόσοι από αυτούς έχουν επιτευχθεί.

Η *ανάδραση* που παρέχεται είναι πολύ καλή καθώς επιτρέπει στον κώδικα του χρήστη να εκτελεστεί άμεσα και τα αποτελέσματα του να γίνουν άμεσα ορατά και κατανοητά.

Η *ισορροπία πρόκλησης και δεξιοτήτων* δεν είναι ιδιαίτερα πετυχημένη. Ο χρήστης καλείται να χρησιμοποιήσει μεθόδους κλάσεων χωρίς να υπάρχουν εισαγωγικά μαθήματα που να εξηγούν μεθόδους ή κλάσεις, ωστόσο δεν είναι ιδιαίτερα σημαντικό αν υπολογίσουμε τις μικρές ηλικίες για τις οποίες προορίζεται το παιχνίδι. Επίσης, δεν υπάρχει η δυνατότητα σε προχωρημένους χρήστες να περάσουν επίπεδα, οι προκλήσεις των οποίων θα τους φανούν εύκολες.

Δεν υπάρχει ιδιαίτερη *συγχώνευση δράσης και συνειδητοποίησης*. Το παιχνίδι έχει αρκετά καλά γραφικά και ήχο, ωστόσο αρκετός χρόνος του χρήστη καταναλώνεται στο γράψιμο κώδικα σε άλλη περιοχή του παιχνιδιού, τελείως αποκομμένη από την ατμοσφαιρική σχεδίαση του παιχνιδιού.

Η *συγκέντρωση* διατηρείται σε υψηλά επίπεδα, λόγω των περιορισμένων αλλά πετυχημένων επιλογών που παρέχονται στον χρήστη, καθώς και της ατμοσφαιρικής μουσικής.

Ο *έλεγχος* που παρέχεται κατά την εκπαιδευτική διαδικασία, είναι σε υψηλό βαθμό. Ο χρήστης μπορεί να γράψει κώδικα όποτε θέλει και να δει άμεσα τα αποτελέσματα με οπτικό τρόπο, ενώ μπορεί ακόμη και να μετακινηθεί μπρος και πίσω στον χρόνο στην εκτέλεση των εντολών.

Η *απώλεια της ατομικής συνείδησης* δεν επιτυγχάνεται ιδιαίτερα, καθώς ο χρήστης δεν αισθάνεται ιδιαίτερα απορροφημένος με τον σκοπό του επιπέδου. Αντιλαμβάνεται τον στόχο του, ωστόσο το αίσθημα του εγωϊσμού ή αρνητισμού ενδεχομένως να έρθει στο προσκήνιο, ιδιαίτερα σε προχωρημένους χρήστες ή σε επίπεδα που απαιτούν zoom ή αρκετών δοκιμών και λάθους.

Ο *χρόνος επιβραδύνεται* στο παιχνίδι αυτό, λόγω της συχνής κίνησης του χαρακτήρα μπροστά και πίσω στο επίπεδο, καθώς και των πολλών δοκιμών. Ο χρήστης παρακολουθεί, επαναληπτικά, την ίδια διαδικασία μέχρι να πετύχει τον σκοπό του, κάτι που μεγαλώνει δραστικά τον χρόνο του παιχνιδιού, αλλά προκαλεί και κούραση.

Τέλος, ο χρήστης αισθάνεται ικανοποίηση κατά την εκτέλεση της διαδικασίας, γιατί πρακτικά ο κώδικας που έγραψε ευθύνεται για αυτό. Η *αυτόνομη εμπειρία* επιτυγχάνεται στο παιχνίδι καθώς κάθε επίπεδο είναι αυτόνομο και προσφέρει συγκεκριμένη γνώση.

2.7.4 Κίνητρο

Το *ενδιαφέρον* του χρήστη διατηρείται σε καλό επίπεδο, από τις πρώτες κιόλας εικόνες του παιχνιδιού. Το RPG στήσιμο του παιχνιδιού, η μουσική, η περιεκτική πληροφόρηση, η ελκυστική διαχείριση του GUI και τα πολύ ζωντανά χρώματα μπορούν να διατηρήσουν την ενασχόληση του χρήστη με την εφαρμογή για μεγάλο διάστημα. Ωστόσο, σε επίπεδο εκπαιδευτικής διαδικασίας, χάνονται αρκετοί πόντοι καθώς το παιχνίδι γίνεται αρκετά αργό και κουραστικό, το ενδιαφέρον διατηρείται κυρίως με RPG στοιχεία – το οποίο μπορεί να μην είναι καν ελκυστικό για σοβαρούς χρήστες – και ο χρήστης δεν αισθάνεται ότι μαθαίνει τα βασικά στοιχεία κάποιας γλώσσα προγραμματισμού, απλά ολοκληρώνει αποστολές.

Τα αποτελέσματα της εκπαιδευτικής διαδικασίας έχουν αρκετή *συνάφεια* με τις απαιτήσεις των περισσοτέρων χρηστών (κυρίως αρχάριων), ωστόσο ο τρόπος εκπαίδευσης δεν βοηθάει ιδιαίτερα ώστε να γίνει κατάλληλη σύνδεση με τις πραγματικές απαιτήσεις του πραγματικού κόσμου, κυρίως λόγω επιπέδου. Πολλοί χρήστες θα αισθανθούν ότι το παιχνίδι είναι είτε δύσκολο, είτε εύκολο ή περισσότερο επικεντρωμένο στο ψυχαγωγικό κομμάτι.

Η *αυτοπεποίθηση* δεν ενισχύεται εύκολα. Το παιχνίδι παρέχει ένα έτοιμο API σε κάθε επίπεδο, το οποίο ο εκπαιδευόμενος μπορεί να χρησιμοποιηθεί στον κώδικα του και είναι μέρος των *ικανοτήτων* του. Ωστόσο, ο εκπαιδευόμενος μπορεί να μην ξέρει καν τι είναι οι μέθοδοι και πως ακριβώς υλοποιούνται αυτά που ο ίδιος γράφει στο πλαίσιο σχεδίασης. Επίσης, το γεγονός ότι κάθε επίπεδο επιλύεται (συνήθως) με συγκεκριμένο τρόπο, ο οποίος μάλιστα του υποδεικνύεται οπτικά, δεν βοηθάει στο να χτίσει την αυτοπεποίθηση του. Ωστόσο, όπως αναφέρθηκε και παραπάνω, το παιχνίδι προορίζεται για μικρές ηλικίες και κατανόηση βασικών εννοιών προγραμματισμού και έτσι δεν είναι ιδιαίτερα σημαντική η κατανόηση της υλοποίησης.

Οι εκπαιδευόμενοι νιώθουν *ικανοποίηση* όταν ολοκληρώνουν ένα επίπεδο, ιδιαίτερα εάν αυτό απαιτούσε αρκετές εντολές, ήταν δύσκολο ή μεγάλο σε έκταση. Σε συνδυασμό με τις αρκετές περιοχές του παιχνιδιού, κάθε μια εκ των οποίων απαιτεί διαφορετικό επίπεδο γνώσης και συντελεί στο μεγάλο μέγεθος του παιχνιδιού, η αίσθηση αυτή ενισχύεται ακόμη παραπάνω.

2.7.5 Αποτελέσματα

Τα αποτελέσματα της αξιολόγησης, με βάση το EFM πλαίσιο σχεδίασης, παρουσιάζονται στον Πίνακα 1. Οι βαθμοί κυμαίνονται από 0 (απουσία ή κακή χρήση) μέχρι 5 (πλήρης εφαρμογή ή πολύ καλή χρήση) της κάθε ιδιότητας & χαρακτηριστικού στο EFM πλαίσιο σχεδίασης που επιτυγχάνεται στο παιχνίδι. Τα αποτελέσματα αυτά θα χρησιμοποιηθούν κατά την ανάπτυξη του παιχνιδιού:

Πίνακας 1: CodeCombat - Αποτελέσματα αξιολόγησης

	Αποτελεσματικό μαθησιακό περιβάλλον	**0**	**1**	**2**	**3**	**4**	**5**
1	Βαθμός αλληλεπίδρασης και ανάδρασης				✓		
2	Συγκεκριμένοι στόχοι και διαδικασίες						✓
3	Κίνητρο		✓				
4	Συνεχής αίσθηση της πρόκλησης		✓				
5	Αίσθημα άμεσης αλληλεπίδρασης με το περιβάλλον				✓		
6	Κατάλληλα και σχετικά εργαλεία			✓			
7	Αποφυγή της απόσπασης προσοχής του εκπαιδευομένου			✓			
	Εμπειρία ροής	**0**	**1**	**2**	**3**	**4**	**5**
1	Στόχοι της δραστηριότητας						✓
2	Αναμφισβήτητη ανάδραση					✓	
3	Ισορροπία πρόκλησης και δεξιοτήτων		✓				
4	Συγχώνευση δράσης και συνειδητοποίησης			✓			
5	Συγκέντρωση					✓	
6	Έλεγχος						✓
7	Απώλεια της ατομικής συνείδησης			✓			
8	Μεταμόρφωση του χρόνου		✓				
9	Αυτόνομη εμπειρία					✓	
	Κίνητρο	**0**	**1**	**2**	**3**	**4**	**5**
1	Ενδιαφέρον				✓		
2	Συνάφεια			✓			
3	Αυτοπεποίθηση				✓		
4	Ικανοποίηση					✓	

2.8 ColoBot

2.8.1 Περιγραφή

Το *ColoBot* (http://colobot.info) είναι ένα παιχνίδι σοβαρού σκοπού το οποίο διαδραματίζεται στο διάστημα και αποτελεί προϊόν της εταιρείας ICC & TerraNova Team. Αποτελεί την τεχνολογική εξέλιξη ενός αρκετά παλιότερου τίτλου με όνομα *CeeBot* (2003) της εταιρείας Epsitec SA, για το οποίο αναπτύχθηκαν τέσσερις (4) εκδόσεις του παιχνιδιού, κάθε μια για συγκεκριμένη ομάδα χρηστών, π.χ. το *CeeBot-4* (2004) προοριζόταν για χρήστες 15+ ετών. Το παιχνίδι στοχεύει σε χρήστες 10+ ετών. Υποστηρίζει αρκετές γλώσσες, αλλά όχι την Ελληνική.

Στο παιχνίδι χειριζόμαστε έναν αστροναύτη (Εικόνα 6), καθώς και έναν αριθμό άλλων αντικειμένων (bots), τα οποία μπορούν να προγραμματιστούν από εμάς, χρησιμοποιώντας μια εσωτερική ψευδογλώσσα με όνομα *CBOT*, η οποία έχει αρκετά κοινά με τις γλώσσες C++ και Java. Υπάρχει μεγάλη ποικιλία από προγραμματιζόμενα bots, όπως μεταφορείς, μίνι γερανούς, μετατροπείς υλικών και φορτιστές μπαταριών, κάθε ένας με συγκεκριμένες ικανότητες. Το σενάριο αφορά την εύρεση και αποίκιση ενός νέου πλανήτη για την ανθρωπότητα.

Ο χρήστης καλείται να επιλύσει μια σειρά από προκλήσεις, οι οποίες μπορούν να αντιμετωπιστούν με σωστό προγραμματισμό των bots που έχει κάθε φορά διαθέσιμα ο χρήστης. Κάθε επίπεδο έχει συγκεκριμένους στόχους και μεθόδους επίλυσης τους που σκοπό έχουν την εκμάθηση συγκεκριμένων προγραμματιστικών εννοιών για τον εκπαιδευόμενο.

Το παιχνίδι ξεκινάει με τη δημιουργία / επιλογή λογαριασμού χρήστη και επιλογή του avatar μας. Στη συνέχεια, ερχόμαστε αντιμέτωποι με ένα μεγάλο σύνολο επιλογών και τύπων παιχνιδιού. Υπάρχουν αποστολές, ελεύθερο παιχνίδι, ασκήσεις, προκλήσεις, επίπεδα δημιουργημένα από άλλους χρήστες και μάχες μεταξύ bots, τα οποία θα ικανοποιήσουν τους χρήστες που ψάχνουν διαφορετικές μορφές παιχνιδιού.

Κάθε τύπος παιχνιδιού περιέχει *κεφάλαια* και κάθε κεφάλαιο περιέχει έναν αριθμό *επιπέδων*, τα οποία αποτελούν πρακτικά τις προγραμματιστικές προκλήσεις του παιχνιδιού. Σε κάθε επίπεδο, ο χρήστης μεταφέρεται σε έναν αφιλόξενο πλανήτη, έχοντας έναν στόχο να εκπληρώσει.

Εικόνα 6: ColoBot - Αρχική οθόνη

2.8.2 Αποτελεσματικό μαθησιακό περιβάλλον

Ο *βαθμός αλληλεπίδρασης και ανάδρασης* είναι μεγάλος. Σε κάθε επίπεδο, ο παίχτης είναι πρακτικά μόνος του και μπορεί μόνο να αλληλεπιδράσει με τα διαθέσιμα bots που έχει, γράφοντας κατάλληλο κώδικα σε αυτά. Η άμεση οπτική ανάδραση που επιτυγχάνεται, βοηθάει τον χρήστη να καταλάβει τα λάθη του ή ακόμη περισσότερο να βελτιώσει τον κώδικα του, σε πραγματικό χρόνο.

Σε κάθε επίπεδο υπάρχουν *συγκεκριμένοι στόχοι και διαδικασίες* που πρέπει να πραγματοποιηθούν. Ωστόσο, αυτό δεν γίνεται άμεσα ξεκάθαρο εάν ο χρήστης δεν πατήσει F1 (εμφάνιση των στόχων και σχετικών πληροφοριών επίλυσης τους), κάτι που γίνεται κουραστικό μετά από κάποια επίπεδα.

Το αίσθημα του *κινήτρου* παρέχεται σε αρκετά καλό βαθμό. Το αφιλόξενο περιβάλλον του κόσμου του παιχνιδιού, ο εξαιρετικός χειρισμός του ήρωα και των εργαλείων του (bots), η ιδιαίτερα ομαλή 3D μηχανή γραφικών καθώς και η πληθώρα περιεχομένου κάνει το παιχνίδι αρκετά ενδιαφέρον για να το συνεχίσει

ο χρήστης. Ωστόσο, η επιλογή εκμάθησης μιας ψευδογλώσσας αντί μιας δημοφιλούς γλώσσας του εμπορίου θα αποτρέψει κάποιους να συνεχίσουν ή να ασχοληθούν με το παιχνίδι, αν και δεδομένου των μικρών ηλικιών για τις οποίες προορίζεται το παιχνίδι, αυτό δεν είναι ιδιαίτερα σημαντικό.

Σε εξίσου μεγάλο βαθμό βρίσκεται και η *αίσθηση συνεχούς πρόκλησης*. Οι προκλήσεις των επιπέδων είναι έτσι δομημένες ώστε είτε εκπαιδεύουν τον χρήστη σε νέες προγραμματιστικές έννοιες, είτε χτίζουν σε κάτι που έχει μάθει ήδη. Το επίπεδο δυσκολίας του παιχνιδιού αυξάνεται ισορροπημένα από επίπεδο σε επίπεδο, χωρίς να υπάρχουν πολύ δύσκολα ή πολύ εύκολα επίπεδα.

Το *αίσθημα άμεσης επαφής και αλληλεπίδρασης με το περιβάλλον* είναι επίσης πετυχημένο. Τα επίπεδα στα οποία εκτυλίσσεται η δράση είναι τρισδιάστατα, ρεαλιστικά και η έκταση τους αρκετά μεγάλη. Ο χρήστης μπορεί να μετακινεί τον χαρακτήρα του καθώς και τα bots του οπουδήποτε και με όποιον τρόπο επιθυμεί (με κατάλληλο κώδικα), κάτι που μπορεί να αποβεί ακόμη και μοιραίο εάν ο χρήστης δεν προσέξει.

Το παιχνίδι χρησιμοποιεί *κατάλληλα και σχετικά εργαλεία.* Ο ενσωματωμένος κειμενογράφος που χρησιμοποιείται για σύνταξη κώδικα είναι προσεγμένος και λιτός, χωρίς να μπερδεύει. Τα ηχητικά εφέ είναι απλά αλλά ρεαλιστικά και πετυχημένα ενώ η απουσία μουσικής προσθέτει πόντους στην προσοχή του χρήστη, χωρίς να αφαιρεί από τον παράγοντα διασκέδασης. Η υποστήριξη αρκετών γλωσσών διευρύνει τη βάση χρηστών. Ωστόσο, τα 3D γραφικά του δεν είναι ιδιαίτερα λεπτομερή ενώ τα μοντέλα είναι χοντροκομμένα – υπάρχει όμως ποικιλία μοντέλων και περιβαλλόντων χώρων.

Η *απόσπαση της προσοχής του εκπαιδευομένου* είναι αρκετά δύσκολη. Το παιχνίδι δεν χρησιμοποιεί πολλά οπτικά ή ηχητικά εφέ, ενώ οι στόχοι του είναι σαφείς. Η συνεχής ανάσα του αστροναύτη μας υπενθυμίζει πάντα ότι έχουμε μια σοβαρή δουλειά να διεκπεραιώσουμε.

2.8.3 Εμπειρία ροής

Οι *στόχοι κάθε δραστηριότητας* είναι αρκετά σαφείς και υπάρχει αρκετή πληροφόρηση για την επίλυση τους. Ωστόσο, αρκετές φορές οι στόχοι είναι περίπλοκοι και σχεδόν πάντα ο χρήστης θα πρέπει να ανοίγει τη σελίδα στόχων (ίσως και αρκετές φορές) πριν επιχειρήσει οτιδήποτε.

Η *ανάδραση* του παιχνιδιού παρέχεται με την οπτική συμπεριφορά των bots, τα οποία εμείς προγραμματίζουμε μέσω κώδικα. Μπορούμε άμεσα να καταλάβουμε τι πήγε στραβά ή πως θα βελτιώσουμε κάτι, απλώς παρατηρώντας μέσω της 3D κάμερας την εκτέλεση των ενεργειών ενός bot.

Υπάρχει καλή *ισορροπία πρόκλησης και δεξιοτήτων*. Οι χρήστες καλούνται να επιλύσουν προβλήματα, χρησιμοποιώντας πάντα ένα πεπερασμένο σύνολο *ικανοτήτων* (σε μορφή συναρτήσεων ενός εσωτερικού API). Προχωρημένα επίπεδα απαιτούν περισσότερο κώδικα, κάτι που δικαιολογείται εάν ο χρήστης έχει φτάσει ως εκεί. Ωστόσο, ακόμη και στα εισαγωγικά επίπεδα, δεν γίνεται αναφορά σε βασικές έννοιες όπως μεταβλητές, συναρτήσεις ή κλάσεις, με αποτέλεσμα ο χρήστης να μαθαίνει περισσότερο το API μιας εύκολης 3D μηχανής παρά μια γλώσσα προγραμματισμού.

Η *συγχώνευση δράσης και συνειδητοποίησης* είναι σχετικά πετυχημένη. Ο χρήστης αφοσιώνεται αρκετά εύκολα στην ολοκλήρωση των επιπέδων και αναφέρθηκαν αρκετοί λόγοι για αυτό παραπάνω. Ωστόσο, δεν συνειδητοποιεί εύκολα τις συνέπειες των πράξεων του, κυρίως λόγω της 3D οπτικής του παιχνιδιού. Οι αποστάσεις είναι δύσκολο να μετρηθούν σε τέτοιο περιβάλλον, ενώ κάποιοι στόχοι απαιτούν να συνταχθούν αρκετές γραμμές κώδικα προτού τις εκτελέσουμε, κάτι που οδηγεί σε λάθη και πολλές επαναλήψεις και εν τέλει ένα συνεχές αίσθημα αμφιβολίας για τον κώδικα που γράφουμε.

Η *συγκέντρωση* διατηρείται σε υψηλό επίπεδο. Η απουσία μουσικής και τα επιτυχημένα ηχητικά εφέ, το αχανές και σιωπηλό περιβάλλον, οι συγκεκριμένοι στόχοι καθώς και ο συχνός προγραμματισμός των bots βοηθούν στο χτίσιμο μας επιβλητικής ατμόσφαιρας που καθηλώνει τον χρήστη.

Ο *έλεγχος* του παιχνιδιού βρίσκεται επίσης ψηλά. Η εφαρμογή συνδυάζει αρμονικά τόσο τον διαδικαστικό προγραμματισμό, όσο και έννοιες αντικειμενοστραφούς προγραμματισμού, χρησιμοποιώντας φυσικά αντικείμενα τα οποία βρίσκονται στον χάρτη. Ο εκπαιδευόμενος ελέγχει πλήρως τις ενέργειες του, ενώ δεν υπάρχουν μοναδικές λύσεις στις προκλήσεις που παρουσιάζονται

Η *απώλεια της ατομικής συνείδησης* επιτυγχάνεται σε μέτριο βαθμό, κυρίως λόγω της συγκέντρωσης που αναφέρθηκε παραπάνω. Ο χρήστης δεν αισθάνεται να πιέζεται ή να πλήττει έξω από επιτρεπτά όρια και τα επίπεδα είναι έτσι δομημένα ώστε να διαφέρουν μεταξύ τους. Ωστόσο, αρκετή ώρα θα περάσει ο εκπαιδευόμενος στον ενσωματωμένο κειμενογράφο, γράφοντας κώδικα ή στην

σελίδα οδηγιών, κάτι που είτε καλύπτει όλη την οθόνη είτε σταματάει τα ηχητικά εφέ, επηρεάζοντας αρνητικά την ατμόσφαιρα του παιχνιδιού και οδηγώντας κάποιες φορές σε κλείσιμο του παιχνιδιού.

Ο χρήστης χάνει αρκετές φορές *την αίσθηση του χρόνου*, ιδιαίτερα όταν προγραμματίζει πετυχημένα ένα από τα bots του και το βλέπει να πετυχαίνει τον στόχο του. Ωστόσο, όταν ο χρήστης αναγκάζεται να διαβάσει τη σελίδα οδηγιών, ο χρόνος κυλάει εξαιρετικά αργά, κυρίως λόγω του μεγάλου μεγέθους πληροφοριών, της δυσκολίας τους καθώς και της αίσθησης ότι διαβάζει ιστοσελίδα – κάτι που δεν απέχει από την αλήθεια, λόγω της παρουσίας links.

Η *αυτόνομη εμπειρία* είναι πολύ πετυχημένη. Κάθε επίπεδο έχει έναν και μοναδικό στόχο, ο οποίος δεν είναι άλλος από την εκπαίδευση του χρήστη σε κάποιες προγραμματιστικές έννοιες. Τα επίπεδα και οι προκλήσεις είναι έξυπνα δομημένα, απαιτούν σκέψη και ακριβή προγραμματισμό και ο χρήστης νιώθει ιδιαίτερη ικανοποίηση όταν τα βλέπει όλα σε λειτουργία όταν πατήσει το κουμπί της εκτέλεσης.

2.8.4 Κίνητρο

Το *ενδιαφέρον* του χρήστη προκαλείται από την πρώτη κιόλας οθόνη του παιχνιδιού, παραπέμποντας περισσότερο σε hacking παιχνίδι. Οι πολλοί τύποι παιχνιδιών, το μεγάλο περιεχόμενο, καθώς και οι πολλές ρυθμίσεις, ιδιαίτερα των γραφικών, δίνουν την εντύπωση ενός εμπορικού τίτλου, προδιαθέτοντας ευχάριστα για τη συνέχεια. Στο παιχνίδι, ο χρήστης γρήγορα αντιλαμβάνεται την σοβαρή δουλειά που έχει γίνει με τα γραφικά, τις κάμερες, τον εξαιρετικό χειρισμό και τη μηχανή παιχνιδιού γενικότερα.

Η *συνάφεια* του παιχνιδιού είναι σχετικά καλή. Το παιχνίδι καταφέρνει να εκπαιδεύσει τους χρήστες σε πολλές προγραμματιστικές έννοιες. Ωστόσο, η γλώσσα που επέλεξε να το πράξει είναι μια ψευδογλώσσα, η οποία μοιάζει με την C++. Αυτό θα απομακρύνει κάποιους χρήστες, είτε αυτούς που ξέρουν προγραμματισμό σε κάποιον βαθμό και θα ήθελαν να προχωρήσουν, είτε αυτούς που δεν ξέρουν και θα ήθελαν να ξεκινήσουν με κάποια δημοφιλή γλώσσα του εμπορίου. Αυτό έχει αρνητική επίδραση στις απαιτήσεις τους, στο παραγόμενο αποτέλεσμα αλλά και στην σύνδεση του με τις απαιτήσεις του πραγματικού κόσμου.

Όπως αναφέρθηκε και παραπάνω, το παιχνίδι χωρίζει το μαθησιακό του περιεχόμενο σε κεφάλαια, με κάθε κεφάλαιο να περιέχει έναν αριθμό επιπέδων. Κάθε επίπεδο έχει ως στόχο είτε να εκπαιδεύσει τον χρήστη σε μια νέα προγραμματιστική έννοια, είτε να χτίσει σε μια υπάρχουσα. Ο χρήστης αντιλαμβάνεται ότι κάθε επίπεδο θα αξιοποιήσει τις υπάρχουσες γνώσεις του και ενδεχομένως να τον εκπαιδεύσει σε κάτι καινούργιο. Αυτό το γεγονός τον προδιαθέτει με ένα δυνατό αίσθημα *αυτοπεποίθησης* για τη συνέχεια. Ωστόσο, όπως αναφέρθηκε και παραπάνω, αρκετές φορές ο χρήστης θα χρειαστεί να ανατρέξει στην σελίδα οδηγιών, καθώς μόνο εκεί αναλύονται οι στόχοι και το API γενικότερα σε βάθος, κάτι που μεταβάλλει το αίσθημα αυτοπεποίθησης.

Τέλος, το αίσθημα της *ικανοποίησης* επιτυγχάνεται σε μεγάλο βαθμό. Οι προκλήσεις είναι ισορροπημένες με το τρέχον επίπεδο του εκπαιδευόμενου, ούτε εύκολες αλλά ούτε και δύσκολες και όπως αναφέρθηκε και παραπάνω, ο χρήστης ξέρει ότι κάθε γνώση που αποκομίζει θα χρησιμοποιηθεί στα αμέσως επόμενα επίπεδα.

2.8.5 Αποτελέσματα

Τα αποτελέσματα της αξιολόγησης, με βάση το EFM πλαίσιο σχεδίασης, παρουσιάζονται στον Πίνακα 2. Οι βαθμοί κυμαίνονται από 0 (απουσία ή κακή χρήση) μέχρι 5 (πλήρης εφαρμογή ή πολύ καλή χρήση) της κάθε ιδιότητας & χαρακτηριστικού στο EFM πλαίσιο σχεδίασης που επιτυγχάνεται στο παιχνίδι. Τα αποτελέσματα αυτά θα χρησιμοποιηθούν κατά την ανάπτυξη του παιχνιδιού:

Πίνακας 2: ColoBot - Αποτελέσματα αξιολόγησης

	Αποτελεσματικό μαθησιακό περιβάλλον	**0**	**1**	**2**	**3**	**4**	**5**
1	Βαθμός αλληλεπίδρασης και ανάδρασης					✓	
2	Συγκεκριμένοι στόχοι και διαδικασίες				✓		
3	Κίνητρο					✓	
4	Συνεχής αίσθηση της πρόκλησης					✓	
5	Αίσθημα άμεσης αλληλεπίδρασης με το περιβάλλον						✓
6	Κατάλληλα και σχετικά εργαλεία				✓		
7	Αποφυγή της απόσπασης προσοχής του εκπαιδευομένου					✓	
	Εμπειρία ροής	**0**	**1**	**2**	**3**	**4**	**5**
1	Στόχοι της δραστηριότητας				✓		
2	Αναμφισβήτητη ανάδραση					✓	
3	Ισορροπία πρόκλησης και δεξιοτήτων				✓		
4	Συγχώνευση δράσης και συνειδητοποίησης			✓			
5	Συγκέντρωση						✓
6	Έλεγχος					✓	
7	Απώλεια της ατομικής συνείδησης			✓			
8	Μεταμόρφωση του χρόνου			✓			
9	Αυτόνομη εμπειρία					✓	
	Κίνητρο	**0**	**1**	**2**	**3**	**4**	**5**
1	Ενδιαφέρον					✓	
2	Συνάφεια			✓			
3	Αυτοπεποίθηση				✓		
4	Ικανοποίηση					✓	

3 Σχεδιασμός του Παιχνιδιού

3.1 Εισαγωγή

Στην εργασία αυτή θα σχεδιαστεί ένα παιχνίδι σοβαρού σκοπού. Ο σχεδιασμός ενός παιχνιδιού περιλαμβάνει μια σειρά από διαδικασίες οι οποίες θα καθορίσουν το σενάριο και τον σκοπό του, τις τεχνολογίες με τις οποίες θα δημιουργηθεί, τον τρόπο με τον οποίο θα παίζεται (gameplay), σε ποια συστήματα θα μπορεί να παιχτεί κλπ.

Οι διαδικασίες αυτές θα περιγραφούν αρχικά στο Game Design Document το οποίο ακολουθεί. Στο έγγραφο αυτό θα περιγραφούν οι απαιτήσεις του παιχνιδιού και, αν χρειαστεί, θα παρουσιαστεί αναθεωρημένη έκδοση του στο τέλος της διαδικασίας σχεδίασης.

Η διαδικασία σχεδίασης που θα ακολουθήσει δεν θα διαφέρει από αυτή ενός ολοκληρωμένου βιντεοπαιχνιδιού για υπολογιστές. Κάθε παιχνίδι εμπεριέχει ένα σενάριο και έναν σκοπό, συγκεκριμένο gameplay, τις τεχνολογίες στις οποίες στηρίζεται για να αποδώσει γραφικά, ήχο και χειρισμό, καθώς και το περιεχόμενο του όπως γραφικά, ήχος κλπ.

Ωστόσο, με την εργασία αυτή δημιουργείται ταυτόχρονα και ένα παιχνίδι το οποίο θα είναι *σοβαρού σκοπού*. Έτσι, θα υπάρχει μέριμνα για το εκπαιδευτικό περιεχόμενο και τον τρόπο με τον οποίο θα γίνει εισαγωγή του περιεχομένου αυτού στο παιχνίδι, καθώς και τους τρόπους με τους οποίους θα αλληλεπιδράει ο χρήστης – εκπαιδευόμενος με αυτό ώστε να πετυχαίνει τον βασικό σκοπό της η εφαρμογή, αυτόν της εκπαίδευσης του χρήστη.

Για να επιτευχθεί αυτό, χρησιμοποιείται ένα συγκεκριμένο πλαίσιο σχεδίασης, το *EFM* (Song, & Zhang, 2008), το οποίο περιγράφεται αναλυτικά στο προηγούμενο κεφάλαιο. Συνοπτικά, το πλαίσιο αυτό ορίζει ένα σύνολο ιδιοτήτων & χαρακτηριστικών τα οποία θα πρέπει να ακολουθήσει το παιχνίδι έτσι ώστε να πετύχει τον εκπαιδευτικό σκοπό του, παρέχοντας ταυτόχρονα τους κατάλληλους κινητήριους μηχανισμούς ώστε να προκαλεί ενδιαφέρον χωρίς να κουράζει κατά την εκπαίδευση.

3.2 Game Design Document

Στον Πίνακα 3, περιγράφονται οι πληροφορίες και οι τεχνικές απαιτήσεις του παιχνιδιού. Κατά τη σχεδίαση και υλοποίηση της εφαρμογής, ενδέχεται κάποιες από τις απαιτήσεις ή πληροφορίες να αλλάξουν. Εάν χρειαστεί, θα υπάρξει αναθεώρηση του Game Design Document στο τέλος. Τέλος, οι ερωτήσεις του συγκεκριμένου Game Design Document προέρχονται από το Australian STEM Video Game Challenge (https://stemgames.org.au), ενός διαγωνισμού ο οποίος ενισχύει φοιτητές της Αυστραλίας με επιστημονική, τεχνολογική, μηχανική και μαθηματική κατεύθυνση και έχουν μεταφραστεί στα Ελληνικά για τις ανάγκες της διπλωματικής εργασίας.

Πίνακας 3: Game Design Document του παιχνιδιού

Πληροφορίες ομάδας	
Όνομα	Ελευθεριάδης Σάββας
Επισκόπηση παιχνιδιού	
Τίτλος παιχνιδιού Πως θα ονομάζεται το παιχνίδι; Πως το όνομα του παιχνιδιού θα βοηθήσει πιθανούς παίχτες να αναγνωρίσουν πως περίπου θα είναι το παιχνίδι;	Το παιχνίδι θα λέγεται **Office Madness**. Το όνομα αυτό δίνει μια ιδέα της φύσης του παιχνιδιού, ότι πρόκειται δηλαδή για κάποιου είδους εξομοίωση σε εργασιακό περιβάλλον. Όσοι αρέσκονται σε εξομοιωτές, ο σχετικά αφηρημένος τίτλος θα τους κεντρίσει πιθανώς το ενδιαφέρον.
Περιγραφή παιχνιδιού Σκεφτείτε το ως άσκηση marketing: Πουλήστε το παιχνίδι στον αναγνώστη – περί τίνος πρόκειται; Τι είδους παιχνίδι είναι; Για ποιον είναι;	Το παιχνίδι αφορά εξομοίωση εργασίας γραφείου ενός προγραμματιστή κάποιας εταιρείας λογισμικού. Στοχεύει σε χρήστες που έχουν τουλάχιστον βασικές γνώσεις προγραμματισμού της γλώσσας C++. Ο τύπος παιχνιδιού είναι job & programming simulator. Στο παιχνίδι υποδυόμαστε τον ρόλο ενός προγραμματιστή σε μια εταιρεία λογισμικού, στον οποίο ανατίθενται διάφορες προγραμματιστικές εργασίες
Κοινό Για ποιον σχεδιάζεται το παιχνίδι; Προορίζεται ειδικά για παιδιά; Ενήλικες; Όλες τις ηλικίες;	Το παιχνίδι θα σχεδιαστεί ώστε να το παίξουν ερασιτέχνες ή χομπίστες προγραμματιστές, οι οποίοι έχουν γνώση αντικειμενοστραφών γλωσσών προγραμματισμού. Οπότε απαιτούνται τουλάχιστον βασικές γνώσεις και ηλικίες από 12+. Επειδή αφορά προσομοίωση σε εργασιακό περιβάλλον και απαιτεί σχετική σοβαρότητα, δεν θα βάζαμε ανώτατο όριο ηλικίας.
Χαρακτήρες / Ρόλοι Ποιος / ποιοι είναι οι κύριοι χαρακτήρες στο παιχνίδι σας; Ποιον ρόλο παίζουν στην ιστορία; Ποιο είναι το κίνητρο των χαρακτήρων αυτών στο παιχνίδι;	Όπως αναφέρθηκε και πριν, το παιχνίδι είναι ένας προσομοιωτής εργασίας γραφείου ενός προγραμματιστή σε μια εταιρεία λογισμικού. Ο πρωταγωνιστής είναι ένας προγραμματιστής και – αρχικά – θα παίζει τον ρόλο ενός προγραμματιστή entry level, δηλαδή με μικρή εμπειρία. Σκοπός του θα είναι να επιλύει τα όποια προγραμματιστικά

	προβλήματα του ανατίθενται, ενώ το κίνητρο του θα είναι μια καλύτερη θέση στην εταιρεία π.χ. προαγωγή σε junior developer, όπου οι προκλήσεις αλλά και οι ευθύνες θα είναι αυξημένες.
Περιβάλλον Σε τι χώρο διαδραματίζεται το παιχνίδι; Κάτω από τι συνθήκες; Έχουν οι συνθήκες αυτές οποιαδήποτε επίδραση στο gameplay που ίσως χρειαστεί να σκεφτούμε;	Το παιχνίδι διαδραματίζεται σε κάποιο γραφείο της εταιρείας λογισμικού που εργάζεται ο προγραμματιστής, σε συνθήκες πραγματικής εργασίας. Οι συνθήκες αυτές επιδρούν στο gameplay καθώς θα υπάρχουν χρονικοί περιορισμοί, υπό τη μορφή ωραρίου εργασίας, στις εργασίες που μας έχουν ανατεθεί, κάτι που θα κρίνει την εργασιακή μας εξέλιξη στην εταιρεία. Γενικά, όλο το παιχνίδι βασίζεται στο “περιβάλλον” και το πως ο προγραμματιστής θα καταφέρει να το ελέγξει.
Gameplay / Διαδικασίες	
Στόχοι Τι είδους παιχνίδι δημιουργείτε; Ποιος είναι ο στόχος του παιχνιδιού; Τι προσπαθεί να επιτύχει ο χρήστης;	Ο σκοπός του παιχνιδιού είναι να πρσομοιώσει τις εργασιακές συνθήκες σε μια εταιρεία λογισμικού (μέσα στα πλαίσια της διασκέδασης πάντα), εκπαιδεύοντας ταυτόχρονα τους χρήστες σε προγραμματιστικές έννοιες μέσω ιδιαίτερων διαδικασιών που θυμίζουν mini games. Ο χρήστης θα προσπαθεί να επιλύει προγραμματιστικές προκλήσεις και ως αντάλλαγμα θα μπορεί να εξελιχθεί στην εταιρεία και – πιθανόν – σε RPG στατιστικά π.χ. ικανότητα κώδικα, ταχύτητα, αλγόριθμοι κλπ.
Προοπτική Ποια θα είναι η προοπτική του παίχτη όσο παίζει το παιχνίδι; Θα βιώσουν την εμπειρία από πρώτου προσώπου κάμερα; Από πλάγια (όπως ένα platform παιχνίδι;) Ή από top-down προοοπτική; Θα είναι 2 διαστάσεων (2D) ή 3 διαστάσεων (3D) παιχνίδι;	Η προοπτική του παιχνιδιού θα είναι ψευδο-3D. Θα χρησιμοποιηθεί ένα 2D υπόβαθρο και 2D sprites, τα οποία όμως θα δίνουν την εντύπωση 3D. Η προοπτική δεν θα διαφέρει από ένα top-down RPG παιχνίδι, όπως πχ. Το Zelda. Τα 2D sprites θα αφορούν αντικείμενα ενός γραφείου προγραμματιστή. Όλο το παιχνίδι θα είναι 2D και θα καταλαμβάνει όλη την οθόνη (fullscreen).
Χειρισμός Πως θα παίζουν ή θα αλληλεπιδρούν οι παίχτες με το παιχνίδι; Πως θα είναι ο χειρισμός; Πως θα λειτουργούν;	Οι χρήστες θα αλληλεπιδρούν με το περιβάλλον και τον προγραμματιστή κυρίως μέσω του ποντικιού αλλά και του πληκτρολογίου – π.χ. για σύνταξη κώδικα –. Η επιλογή των προγραμματιστικών εργασιών και του μενού γενικότερα θα γίνεται μέσω του ποντικιού.
Ιδέα / Πρωτοτυπία Υπάρχουν άλλα παιχνίδια με παρόμοιο gameplay; Παρόμοια λειτουργικότητα; Παρόμοιες ιστορίες ή χαρακτήρες; Πως θα διαφέρει το παιχνίδι σας; Γιατί οι παίχτες να επιλέξουν το παιχνίδι σας αντί των άλλων; Είναι το παιχνίδι σας αρκετά διαφορετικό ώστε να αξίζει να το φτιάξετε; Γιατί ή γιατί όχι;	Υπάρχουν λίγα εμπορικά παιχνίδια του είδους, για παράδειγμα το βραβευμένο παιχνίδι *Papers, Please* που έχει κυκλοφορήσει στο Steam έχει παρόμοιο gameplay, αλλά δεν είναι σοβαρού σκοπού. Παρόμοιο είναι και το *Not Tonight*, επίσης στο Steam. Υπάρχει το *Job Simulator*, επίσης στο Steam αλλά αφορά απλά μια διασκεδαστική παρωδία σε εργασιακές συνθήκες. Στα λίγα παιχνίδια που υπάρχουν, εκτιμούμε ότι το παιχνίδι δεν θα διαφέρει ιδιαίτερα στον διασκεδαστικό τομέα. Ωστόσο, στον εκπαιδευτικό τομέα, εκτιμούμε ότι χρήστες που ασχολούνται με την τεχνολογία θα το δουν με μεγαλύτερο ενδιαφέρον, ίσως και από περιέργεια. Πιστεύουμε ότι έχει

	μεγάλη αξία η δημιουργία ενός τέτοιου παιχνιδιού, ίσως και από στατιστικής πλευράς (π.χ. ηλικία, εμπειρία, επάγγελμα) σε συνδυασμό με την αποτελεσματικότητα που θα δείξει.
Τεχνικές απαιτήσεις	
Πλατφόρμα Σε τι περιβάλλον θα τρέχει;	Το παιχνίδι θα τρέχει σε περιβάλλον Windows ενώ θα είναι portable και σε άλλα λειτουργικά συστήματα.
Περιβάλλον ανάπτυξης Τι τεχνολογίες θα χρησιμοποιήσετε για να αναπτύξετε το παιχνίδι σας;	Το παιχνίδι θα αναπτυχθεί σε περιβάλλον Windows, κάνοντας χρήση των παρακάτω τεχνολογιών: - C/C++ με compiler MinGW (γλώσσα & compiler) - OpenGL (γραφικά) - OpenAL Soft (ήχος) - Notepad++ (συντάκτης κώδικα)
Απαιτήσεις συστήματος Τι προδιαγραφές συστήματος ή περιφερειακά θα χρειάζεται ο τελικός χρήστης ώστε να παίξει το παιχνίδι σας;	Το παιχνίδι θα απαιτεί μια κάρτα γραφικών συμβατή με OpenGL 2.0+. Θα απαιτεί την ύπαρξη ποντικιού / trackpad, κάρτας ήχου και λίγων μόλις MBs στον σκληρό δίσκο. Με το σημερινό hardware, όλοι οι υπολογιστές θα είναι ικανοί να τρέξουν με ευκολία το παιχνίδι.
Δεξιότητες Τι είδους δεξιοτήτων ή ικανοτήτων θα απαιτηθούν;	Το παιχνίδι θα αναπτυχθεί με εργαλεία cross-platform, χωρίς ιδιαίτερη εξάρτηση από κάποιο IDE ή λειτουργικό σύστημα. Ως εφαρμογή σοβαρού σκοπού η οποία θα δημιουργηθεί με την C++, θα απαιτηθούν αρκετές γνώσεις στην γλώσσα C++. Θα απαιτηθούν επίσης βασικές γνώσεις scripting, compiling και text-editing. Κατά την υλοποίηση του παιχνιδιού, θα απαιτηθούν βασικές γνώσεις επεξεργασίας γραφικών, αρχείων ήχου, ενδεχομένως και διαχείρισης βάσης δεδομένων. Τέλος, επειδή το παιχνίδι θα βασιστεί στο OpenGL, θα απαιτηθούν επίσης βασικές γνώσεις θεωρίας 3D.
Παρουσίαση / Γραφικά	
Στυλ Πως θα παρουσιάζεται το παιχνίδι σε γραφικό επίπεδο; Ποιο θα είναι το βασικό του γραφικό στυλ;	Όπως αναφέρθηκε και παραπάνω, το παιχνίδι διαδραματίζεται σε ένα γραφείο προγραμματιστή μιας εταιρείας λογισμικού. Το γραφείο και γενικότερα όλα τα γραφικά στοιχεία θα εμφανίζονται σε μια μορφή top-down (με κάποια κλίση) ώστε να δίνουν την εντύπωση 3D. Κάποια από τα γραφικά στοιχεία είτε θα περιέχουν animation είτε θα αλλάζουν μορφή με βάση τις ενέργειες του χρήστη, αποτελώντας τον κύριο εκφραστή της αλληλεπίδρασης με τον χρήστη.
Διαδικασία Πως θα επιτύχετε το επιθυμητό σας οπτικό στυλ; Πως θα φτάσετε από το στάδιο της ιδέας στο ολοκληρωμένο προϊόν;	Για να επιτευχθεί η επιθυμητή εμφάνιση του παιχνιδιού, θα χρησιμοποιηθούν κάποια έτοιμα υπόβαθρα (backgrounds) και sprites που θα αφορούν αντικείμενα γραφείου. Τα στοιχεία αυτά θα επεξεργαστούν κατάλληλα σε γραφικό επίπεδο ώστε να μπορούν να γίνουν μέρος της οθόνης. Σε αρχικό επίπεδο, θα γίνουν κάποια mockups των γραφικών στοιχείων, καθώς και όλων των οθονών του παιχνιδιού. Κάθε γραφικό στοιχείο θα δουλεύεται ξεχωριστά, μέχρι να ικανοποιεί με την εμφάνιση και τον τρόπο λειτουργίας του.

3.3 Αποτελέσματα αξιολόγησης

Στο προηγούμενο κεφάλαιο πραγματοποιήθηκε ανάλυση σε μια σειρά παιχνιδιών σοβαρού σκοπού, με σκοπό να αξιολογηθούν στο πλαίσιο σχεδίασης EFM. Τα αποτελέσματα της αξιολόγησης κρίνονται ιδιαιτέρως χρήσιμα καθώς θα αναδείξουν τους τομείς στους οποίους τα παιχνίδια αυτά τα πήγαν πολύ καλά ή πολύ άσχημα, μια γνώση που θα αξιοποιηθεί κατά τη σχεδίαση των μηχανισμών του παιχνιδιού.

Η προσοχή θα εστιαστεί μόνο σε ιδιότητες & χαρακτηριστικά του πλαισίου σχεδίασης EFM, στα οποία τα παιχνίδια που αξιολογήθηκαν τα πήγαν περίφημα ή απέτυχαν. Συγκεκριμένα, θα γίνει αναφορά μόνο στις ιδιότητες του πλαισίου σχεδίασης EFM, στις οποίες τα παιχνίδια:

- ✔ πέτυχαν από **0 - 1** βαθμούς (αποτυχία)
- ✔ πέτυχαν από **4 - 5** βαθμούς (επιτυχία)

, αγνοώντας πρακτικά τους βαθμούς **2** & **3** στις ιδιότητες ως άνευ σημασίας, καθώς το παιχνίδι ούτε πέτυχε ούτε απέτυχε στον συγκεκριμένο τομέα. Στη συνέχεια, θα συγκεντρωθούν σε πίνακες οι ιδιότητες αυτές του πλαισίου EFM για κάθε παιχνίδι και θα αναφερθούν τρόποι αντιμετώπισης των προβλημάτων που εντοπίστηκαν κατά την αξιολόγηση των παιχνιδιών, προσαρμοσμένοι στις ανάγκες του προς ανάπτυξη παιχνιδιού. Τέλος, αυτοί οι τρόποι αντιμετώπισης θα αξιοποιηθούν παρακάτω, στο σχεδιασμό των μηχανισμών του παιχνιδιού.

3.3.1 CodeCombat

Στον Πίνακα 4, καταγράφονται οι σημαντικότερες ιδιότητες του πλαισίου σχεδίασης EFM, στις οποίες το παιχνίδι παρουσίασε ενδιαφέρον κατά την αξιολόγηση του, είτε αρνητικά είτε θετικά. Τέλος, προτείνονται τρόποι αντιμετώπισης των αποτελεσμάτων που προκάλεσαν οι ιδιότητες αυτές, οι οποίοι θα υλοποιηθούν κατά την ανάπτυξη της παρούσας εφαρμογής.

Πίνακας 4: CodeCombat - Αξιοποίηση αποτελεσμάτων αξιολόγησης

Αποτελεσματικό μαθησιακό περιβάλλον	**Αντιμετώπιση**
Συγκεκριμένοι στόχοι και διαδικασίες (5/5) ✓ Ξεκάθαρη εικόνα στόχων, άμεσα ορατοί.	✓ Συγκεκριμένοι και άμεσα ορατοί στόχοι σε κάθε επίπεδο.
Κίνητρο (1/5) × Έλλειψη σεναρίου, αδύναμη ατμόσφαιρα.	✓ Ρεαλιστικό σενάριο.
Συνεχής αίσθηση της πρόκλησης (1/5) × Πολλές πληροφορίες, επίδραση gameplay.	✓Απλό GUI στα πρώτα επίπεδα, πιο τεχνικό και με περισσότερους μηχανισμούς σε επόμενα επίπεδα.
Εμπειρία ροής	**Αντιμετώπιση**
Στόχοι της δραστηριότητας (5/5) ✓ Ξεκάθαροι στόχοι επιπέδων, άμεσα ορατοί.	✓ Συγκεκριμένοι και άμεσα ορατοί στόχοι σε κάθε επίπεδο.
Αναμφισβήτητη ανάδραση (4/5) ✓ Άμεση εκτέλεση κώδικα & αποτελέσματος.	✓ Εκτέλεση κώδικα χρήστη σε compiler (ενσωματωμένο / διαδικτυακό), άμεση ανάδραση μέσω γραφικών μηνυμάτων.
Ισορροπία πρόκλησης και δεξιοτήτων (1/5) × Απουσία μαθημάτων σε βασικές έννοιες, μη επιλογή επιπέδων για προχωρημένους.	✓ Διαφορετικές θέσεις ευθύνης στην εταιρεία π.χ. entry level, junior developer. Ύπαρξη εισαγωγικών μαθημάτων στην αρχή. Αρχική επιλογή θέσης (με τεστ).
Συγκέντρωση (4/5) ✓ Επιλογές, ατμοσφαιρική μουσική.	✓ Λίγες αλλά πρακτικές επιλογές, που θα αυξάνουν με το επίπεδο του χρήστη. Ήχοι εξομοίωσης σε περιβάλλον γραφείου.
Έλεγχος (5/5) ✓ Άμεση σύνταξη κώδικα & ανάδραση.	✓ Χρήση εικονικού Η/Υ και συντάκτη κώδικα.
Μεταμόρφωση του χρόνου (1/5) × Συχνή επανάληψη στην εκτέλεση κώδικα.	✓ Η εκτέλεση κώδικα & ο μηχανισμός ανάδρασης του θα συμβαίνει μια φορά.
Αυτόνομη εμπειρία (4/5) ✓ Ικανοποίηση λόγω κύριας ευθύνης στη σύνταξη κώδικα	✓ Υπευθυνότητα χρήστη για σύνταξη κώδικα, θέση ευθύνης, εξέλιξη θέσης.
Κίνητρο	**Αντιμετώπιση**
Ικανοποίηση (4/5) ✓ Σύνταξη απλού ή περίπλοκου κώδικα, μεγάλο μέγεθος περιεχομένου.	✓ Εικονικός Η/Υ με συντάκτη κώδικα, μαθήματα για entry level μέχρι προβλήματα για senior developer, μεγάλο μέγεθος.

3.3.2 ColoBot

Στον Πίνακα 5, καταγράφονται οι σημαντικότερες ιδιότητες του πλαισίου σχεδίασης EFM, στις οποίες το παιχνίδι παρουσίασε ενδιαφέρον κατά την αξιολόγηση του, είτε αρνητικά είτε θετικά. Τέλος, προτείνονται τρόποι

αντιμετώπισης των αποτελεσμάτων που προκάλεσαν οι ιδιότητες αυτές, οι οποίοι θα υλοποιηθούν κατά την ανάπτυξη της παρούσας εφαρμογής.

Πίνακας 5: ColoBot - Αξιοποίηση αποτελεσμάτων αξιολόγησης

Αποτελεσματικό μαθησιακό περιβάλλον	**Αντιμετώπιση**
Βαθμός αλληλ/δρασης και ανάδρασης (4/5) ✓ Περιβάλλον, άμεση οπτική ανάδραση.	✓ Άμεση αλληλεπίδραση χρήστη με το γραφείο π.χ. σημειώσεις, οθόνη. Ανάδραση μέσω Η/Υ, εικονικές εκτυπώσεις κλπ.
Κίνητρο (4/5) ✓ Περιβάλλον, χειρισμός, περιεχόμενο.	✓ Περιβάλλον με προκλήσεις, εύκολος και ελκυστικός χειρισμός, μεγάλο περιεχόμενο
Συνεχής αίσθηση της πρόκλησης (4/5) ✓ Ισορροπημένη διαβάθμιση δυσκολίας.	✓Διάφορες θέσεις εργασίας στην εταιρεία π.χ. entry level, junior developer κλπ.
Αίσθημα άμεσης αλλ/σης με περιβάλλον (5/5) ✓ Αλλ/δραση με το φυσικό περιβάλλον.	✓ Εξομοίωση εργασιακού περιβάλλοντος με χρήση εικονικού Η/Υ, σύνταξης κώδικα, εκτύπωσης, ωραρίου, εξέλιξης κλπ.
Αποφυγή απόσπασης προσοχής (4/5) ✓ Λιτό GUI, πετυχημένα οπτικά / ηχητικά εφέ.	✓ Κάθε δραστηριότητα είναι σχεδιασμένη για εκπαίδευση αλλά και για παροχή κινήτρου.
Εμπειρία ροής	**Αντιμετώπιση**
Αναμφισβήτητη ανάδραση (4/5) ✓ Άμεση κατανόηση προβληματικού κώδικα λόγω οπτικής ανάδρασης.	✓ Χρήση compiler για άμεση εκτέλεση κώδικα. Άμεση ανάδραση μέσω της εικονικής οθόνης.
Συγκέντρωση (5/5) ✓ Σωστή απόδοση ρεαλισμού λόγω σωστού σχεδιασμού περιβάλλοντος και στόχων.	✓ Ρεαλιστικά ηχητικά και οπτικά εφέ ενός εργασιακού περιβάλλοντος.
Έλεγχος (4/5) ✓ Διαχείριση μηχανημάτων με κώδικα.	✓ Διαχείριση εικονικών εργαλείων ενός γραφείου εργασίας.
Αυτόνομη εμπειρία (4/5) ✓ Εκπαίδευση σε συγκεκριμένες έννοιες.	✓ Τα επίπεδα χωρίζονται σε ημέρες εργασίας , κάθε μια είτε παρέχει εκπαίδευση σε νέες έννοιες είτε χτίζει σε υπάρχουσα γνώση
Κίνητρο	**Αντιμετώπιση**
Ενδιαφέρον (4/5) ✓ Παρουσίαση, επιλογές, περιεχόμενο, σοβαρότητα, 3D μηχανή.	✓ Εισαγωγή στο παιχνίδι, ελκυστικοί οπτικοί μηχανισμοί, ρεαλιστικό σενάριο & στόχοι, εξομοίωση εργασιακού περιβάλλοντος, μεγάλο & συναφές μέγεθος παιχνιδιού.
Ικανοποίηση (4/5) ✓ Ισορροπία δυσκολίας, χτίσιμο γνώσης.	✓ Ισορροπία δυσκολίας με θέσεις εργασίας, συνάφεια με τις πραγματικές απαιτήσεις.

3.4 Το Παιχνίδι

3.4.1 Περιγραφή

Το *Office Madness* είναι ένα παιχνίδι σοβαρού σκοπού το οποίο προσομοιώνει τις συνθήκες εργασίας σε ένα γραφείο μιας εταιρείας λογισμικού. Ο κεντρικός πρωταγωνιστής του παιχνιδιού είναι ένας προγραμματιστής, τον οποίο προσλαμβάνει η εταιρεία, με κύριο σκοπό να ολοκληρώνει τις προγραμματιστικές εργασίες που του ανατίθενται.

Το παιχνίδι είναι ένα υβρίδιο job simulator και εκπαίδευσης στη γλώσσα προγραμματισμού C++. Το παιχνίδι διαδραματίζεται σε ένα γραφείο εργασίας, ενώ ο προγραμματιστής έχει πρόσβαση σε σχετικά εργαλεία της εργασίας του π.χ. οθόνη, εφαρμογές, εργασίες κ.α. Τα επίπεδα έχουν την μορφή *ημερών εργασίας*, όπου η κάθε ημέρα καθορίζεται από συγκεκριμένο ωράριο εργασίας και περιλαμβάνει προγραμματιστικές εργασίες, εκμάθηση προγραμματιστικών εννοιών μέσω κατάλληλα δομημένων οδηγών και άλλες διαδικασίες.

Κύριος σκοπός του παιχνιδιού είναι να εισαγάγει τον χρήστη σε βασικές & προχωρημένες έννοιες της προγραμματιστικής γλώσσας C++, αλλά και να τον εκπαιδεύσει – εικονικά – σε συνθήκες ενός τέτοιου εργασιακού περιβάλλοντος. Με αυτόν τον τρόπο θα υπάρχει άμεση σύνδεση & συνάφεια με το μαθησιακό περιεχόμενο, από την εκπαίδευση μέχρι τη χρήση των γνώσεων αυτών, έστω και εικονικά.

3.4.2 Σκοπός

Στο παιχνίδι, ο βασικός σκοπός είναι η εκπαίδευση των χρηστών στην γλώσσα προγραμματισμού C++. Ένας δευτερεύων σκοπός είναι η εξοικείωση των χρηστών με τις συνθήκες εργασίας σε ένα εικονικό γραφείο μιας εταιρείας λογισμικού, στο οποίο θα αξιοποιούνται – πρακτικά – οι γνώσεις τους, μετά την εκμάθηση τους.

Ο βασικός λόγος που το παιχνίδι σχεδιάστηκε σε τέτοια μορφή είναι για να εμπλουτίσει με τεχνητό *ρεαλισμό* & *υπευθυνότητα* την εφαρμογή, χωρίς ωστόσο να ξεφεύγει από τα επιτρεπτά όρια του παράγοντα *διασκέδαση*.

3.4.3 Ιδέα

Η κύρια έμπνευση για τη σχεδίαση του παιχνιδιού σε αυτή τη μορφή υπήρξαν δυο (2) επιτυχημένα βιντεοπαιχνίδια του εμπορίου, που περιγράφονται παρακάτω. Το κύριο χαρακτηριστικό των παιχνιδιών αυτών υπήρξε ο *ρεαλισμός* καθώς και το αίσθημα *υπευθυνότητας* που πρέπει να επιδείξει ο χρήστης, εάν θέλει να προχωρήσει στο παιχνίδι.

Στα παιχνίδια αυτά, ο κύριος πρωταγωνιστής μπορεί να μείνει χωρίς χρήματα, να φυλακιστεί ή ακόμη και να πεθάνει. Οι ρεαλιστικοί μηχανισμοί, το αίσθημα υπευθυνότητας, η καθημερινή προσπάθεια επιβίωσης, καθώς και οι ελκυστικοί μηχανισμοί gameplay σε μορφή mini-games, καθιστά τα παιχνίδια αυτά εξαιρετικά εθιστικά.

- ✔ **Papers, Please:** Πρόκειται για ένα adventure / puzzle / simulation βιντεοπαιχνίδι στο οποίο ο πρωταγωνιστής παίρνει τον ρόλο ενός ελεγκτή χαρτιών μετανάστευσης (π.χ. διαβατηρίων, εισιτηρίων κ.α.) σε κάποιο κέντρο ελέγχου. Τα επίπεδα του παιχνιδιού χωρίζονται σε ημέρες, όπου κάθε ημέρα που περνάει ο ελεγκτής θα πρέπει να προβαίνει σε όλο και περισσότερους ελέγχους. Στο τέλος δέχεται ή απορρίπτει το αίτημα μετανάστευσης στον πολίτη που έχει απέναντι του. Το παιχνίδι έχει βραβευτεί και έχει κυκλοφορήσει μια ταινία μικρού μήκους για αυτό.
  https://store.steampowered.com/app/239030/Papers_Please
  https://www.youtube.com/watch?v=YFHHGETsxkE

- ✔ **Not Tonight:** Πρόκειται επίσης για ένα adventure / puzzle / simulation βιντεοπαιχνίδι, με στοιχεία RPG, στο οποίο ο πρωταγωνιστής παίρνει τον ρόλο ενός Ευρωπαίου μετανάστη στη Μ. Βρετανία. Στο εναλλακτικό σενάριο του παιχνιδιού, μια ακροδεξιά κυβέρνηση έχει πάρει τον έλεγχο και οι μη-Βρετανοί πολίτες υπόκεινται σε συχνό έλεγχο & είναι υποχρεωμένοι σε καθημερινή εργασία ώστε να ανανεώνεται η visa τους για να συνεχίζουν να ζουν εκεί. Το παιχνίδι είναι βασικά μια εναλλακτική πρόταση του *Papers, Please*, ωστόσο από μια άλλη εταιρεία.
  https://store.steampowered.com/app/733790/Not_Tonight

3.4.4 Είδος

Μέχρι σήμερα, έχουν σχεδιαστεί πολλά παιχνίδια σοβαρού σκοπού ενώ ο αριθμός τους, καθώς και τα πεδία τα οποία αφορούν, αυξάνεται συνεχώς. Ωστόσο, σε αντίθεση με τα εμπορικά βιντεοπαιχνίδια, στα παιχνίδια σοβαρού σκοπού συνηθίζεται να αναφέρεται απλά το πεδίο με το οποίο ασχολούνται π.χ. υγεία, και όχι ο τύπος του παιχνιδιού π.χ. RPG, κάτι που περιορίζει την ταξινόμηση των παιχνιδιών σοβαρού σκοπού.

Μπορεί να υπάρξει μια σύνδεση μεταξύ του μαθησιακού περιεχομένου και των κατάλληλων τύπων παιχνιδιών (Laporte, L., Zaman, B., & Grooff, D.D., 2013), κάτι που θα ενισχύσει την αποτελεσματικότητα του μαθησιακού περιεχομένου. Η έρευνα βασίστηκε στην αξιολόγηση τεσσάρων (4) παιχνιδιών σοβαρού σκοπού, διαφορετικού είδους & μαθησιακού περιεχομένου, ώστε να καταλήξει στο κατάλληλο είδος παιχνιδιών με βάση το μαθησιακό τους περιεχόμενο. Τα αποτελέσματα της έρευνας αυτής συνοψίζονται στην Εικόνα 7:

Learning content and activities											
		Facts				**Behaviours**				**Reasoning**	
		questions	memorization	association	drill	imitation	feedback	coaching	practice	problems	examples
(1) Learning activity useful to convey learning content: health education and programming											
HEALTH	**1** NC	x	x	x	x	x	x	x	x		
	2 N4K	x	x	x	x	x	x	x	x		
PROGRAMMING	**3** CaB									x	x
	4 CoB									x	x
(2) Learning activity provided by genre: simulation, puzzle and strategy/adventure											
SIMULATION	**5** NC					x	x				
PUZZLE	**6** N4K		x	x	x					x	x
	7 CaB		x	x	x					x	x
STRATEGY	**8** CoB						x	x			
		questions	memorization	association	drill	imitation	feedback	coaching	practice	problems	examples
		Facts				**Behaviours**				**Reasoning**	
Learning content and activities											

Εικόνα 7: Αποτελέσματα σύνδεσης τύπων παιχνιδιών & περιεχομένου

Exploring the value of genres in serious games, 2013, p.3

Το παιχνίδι που θα αναπτυχθεί αφορά έναν προσομοιωτή εργασίας, ενώ το μαθησιακό του περιεχόμενο αφορά εκπαίδευση σε βασικές και προχωρημένες έννοιες προγραμματισμού. Σύμφωνα με την έρευνα και τα χαρακτηριστικά της προς ανάπτυξη εφαρμογής, καθορίστηκαν και οι *κατάλληλες* δραστηριότητες για την εφαρμογή, με βάση το μαθησιακό περιεχόμενο & το είδος του, οι οποίες είναι οι ακόλουθες:

- ✔ **Προγραμματισμός:** Προβλήματα & Παραδείγματα
- ✔ **Προσομοίωση:** Μίμηση & Ανάδραση

Στην ενότητα της ανάλυσης μηχανισμών, θα σχεδιαστούν οι κατάλληλες δραστηριότητες στην εφαρμογή, οι οποίες θα αξιοποιούν το παραπάνω αποτέλεσμα ώστε να επιτευχθεί ένα καλύτερο μαθησιακό αποτέλεσμα κατά την εκπαίδευση.

3.4.5 Μηχανισμοί

Στα προηγούμενα κεφάλαια χρησιμοποιήθηκε ένας αριθμός παιχνιδιών σοβαρού σκοπού, από την χρήση των οποίων βγήκαν χρήσιμα συμπεράσματα, αξιοποιώντας τις ιδιότητες του πλαισίου σχεδίασης EFM. Στη συνέχεια, αναλύθηκαν τα αποτελέσματα αυτά, επικεντρώνοντας στα θετικά αλλά και αρνητικά στοιχεία των παιχνιδιών αυτών, αγνοώντας τις ιδιότητες που δεν επηρέασαν ιδιαίτερα.

Αυτό επέτρεψε την απομόνωση των ιδιοτήτων & χαρακτηριστικών του πλαισίου σχεδίασης EFM, στα οποία τα παιχνίδια αυτά είτε πέτυχαν είτε απέτυχαν στο σκοπό τους. Παρακάτω, πραγματοποιείται μια αναλυτική καταγραφή των προτάσεων, με τις οποίες θα αντιμετωπιστούν τα προβλήματα που προέκυψαν κατά την αξιολόγηση των προαναφερθέντων παιχνιδιών, προσαρμοσμένες πάντα στις απαιτήσεις της παρούσας εφαρμογής προς ανάπτυξη. Τέλος, παρουσιάζεται και ένα αρχικό mockup του παιχνιδιού, στην Εικόνα 8:

- ✔ **Μεγάλος βαθμός αλληλεπίδρασης & ανάδρασης:** Ο χρήστης θα έχει άμεσα διαθέσιμα, στο εικονικό του γραφείο, τα εργαλεία για να διεκπεραιώνει τις προγραμματιστικές του αναθέσεις π.χ. την οθόνη Η/Υ. Η χρήση μενού επιλογών θα είναι μειωμένη και έτσι ο χρήστης θα μπορεί, με σχεδόν φυσικό τρόπο, να χρησιμοποιήσει τα εικονικά του εργαλεία, με βάση το ένστικτο του και με άμεση ανάδραση των ενεργειών του π.χ. άνοιγμα οθόνης ή εκτύπωση κώδικα.

- ✔ **Συγκεκριμένοι στόχοι και διαδικασίες:** Κάθε επίπεδο, το οποίο θα έχει την μορφή μιας καθημερινής εργασιακής ημέρας, θα εμφανίζει σε διακριτή περιοχή τους στόχους του επιπέδου, σε μορφή προγραμματιστικών αναθέσεων. Εμφανισιακά, θα έχουν την μορφή σημειώσεων, φύλλων Α4 κλπ. τα οποία θα παρέχονται από την εταιρεία ανά τακτά χρονικά διαστήματα, όσο ο προγραμματιστής διεκπεραιώνει τις προηγούμενες εργασίες του.

- ✔ **Κίνητρο:** Το βασικό σενάριο του παιχνιδιού θα επεξηγείται με κατάλληλο, ατμοσφαιρικό τρόπο (γραφικά & ήχοι εργασιακού περιβάλλοντος) κατά την έναρξη του παιχνιδιού. Ο χρήστης θα αντιλαμβάνεται άμεσα τον σκοπό του ενώ το κίνητρο θα ενισχύεται από την ρεαλιστική ατμόσφαιρα. Ο χειρισμός του παιχνιδιού θα είναι γρήγορος, εύκολος και ελκυστικός, στοχεύοντας τόσο στην εθιστικότητα όσο και στον ρεαλισμό π.χ. άνοιγμα οθόνης Η/Υ, γύρισμα σελίδων στις αναθέσεις, εκτύπωση κώδικα κλπ.

- ✔ **Συνεχής αίσθηση της πρόκλησης:** Στα αρχικά επίπεδα – ημέρες – του παιχνιδιού, το GUI του παιχνιδιού θα είναι πολύ απλό και θα περιορίζεται σε πολύ βασικές λειτουργίες, κάτι που θα βοηθήσει στην εκμάθηση του χειρισμού. Στα επόμενα επίπεδα – ημέρες – του παιχνιδιού, θα προστίθεται κάτι νέο π.χ. νέες γνώσεις για διάβασμα, νέοι τύποι προβλημάτων ή νέες εργασιακές διαδικασίες, οι οποίες θα προκαλούν συνεχώς τον χρήστη, μέσα σε επιτρεπτά πλαίσια πάντα. Τέλος, οι διαφορετικές θέσεις εργασίας στις οποίες θα εξελίσσεται ο χρήστης – υπάλληλος π.χ. junior developer, θα προκαλούν και μόνο με την παρουσία τους, ενώ λειτουργικά θα δυσκολεύουν το μαθησιακό

περιεχόμενο, θα ενισχύουν το αίσθημα ευθύνης και τελικά θα προάγουν τον ρεαλισμό.

- ✔ **Αίσθημα άμεσης αλληλεπίδρασης με το περιβάλλον:** Το περιβάλλον στο οποίο θα εξελίσσεται το παιχνίδι θα είναι μια πλήρη προσομοίωση ενός εργασιακού γραφείου σε μια εταιρεία λογισμικού. Όλες οι ενέργειες θα πραγματοποιούνται μέσω άμεσης αλληλεπίδρασης σε κάποιο εικονικό φυσικό μέσο π.χ. οθόνη Η/Υ, σημειωματάριο με προγραμματιστικές αναθέσεις κ.α.

- ✔ **Αποφυγή απόσπασης της προσοχής του εκπαιδευομένου:** Κάθε δραστηριότητα θα είναι έτσι σχεδιασμένη ώστε να απαιτεί συγκεκριμένες ενέργειες & μικρό χρόνο για την επίλυση της. Αυτό θα μετατρέψει το παιχνίδι σε μια σειρά από *επαναλαμβανόμενες* ενέργειες, με τις οποίες όμως ο χρήστης – υπάλληλος θα αποκτήσει οικειότητα και θα υπάρχει δυσκολία απόσπασης της προσοχής. Τέλος, θα υπάρχουν περιορισμένες επιλογές στο GUI γενικότερα, απαραίτητες κυρίως για την επίλυση των προγραμματιστικών αναθέσεων.

- ✔ **Αναμφισβήτητη ανάδραση:** Στο εικονικό γραφείο εργασίας, θα υπάρχει ένας Η/Υ καθώς και ένας συντάκτης κώδικα και ο οποίος θα εκτελεί τον κώδικα είτε σε έναν – ενσωματωμένο – εκτελέσιμο compiler είτε σε διαδικτυακό compiler. Η ανάδραση θα είναι κατάλληλη και άμεση καθώς η έξοδος του προγράμματος θα αποτυπώνεται στην οθόνη.

- ✔ **Ισορροπία πρόκλησης και δεξιοτήτων:** Το παιχνίδι θα χωρίζει τα επίπεδα σε ημέρες εργασίας, ενώ ο ίδιος ο υπάλληλος – χρήστης θα κατέχει μια συγκεκριμένη θέση στην εταιρεία, η οποία αντανακλά τον βαθμό δυσκολίας και ευθύνης στην εταιρεία. Στην αρχή, ο χρήστης θα είναι απλά ένας entry level προγραμματιστής και οι εργασίες που θα του ανατίθενται, καθώς και οι γνώσεις που θα του παρέχονται θα αφορούν βασικές έννοιες της γλώσσας προγραμματισμού C++. Όσο περνάνε οι μέρες εργασίας, θα προστίθενται νέες γνώσεις, ενώ θα αυξάνεται και ο βαθμός δυσκολίας των εργασιών που θα του ανατίθενται. Κάθε θέση

εργασίας στην εταιρεία π.χ. entry level, junior level κλπ. θα σηματοδοτεί αλλαγή στο περιεχόμενο, στις ευθύνες και εν τέλει στις απαιτήσεις του χρήστη. Τέλος, θα υπάρχει επιλογή στην αρχή για αίτηση θέσης διαφορετική του entry level (για προχωρημένους χρήστες).

- ✔ **Συγκέντρωση:** Το παιχνίδι, αρχικά, θα έχει λίγες επιλογές οι οποίες και θα είναι άμεσα συνδεδεμένες, είτε με το εκπαιδευτικό περιεχόμενο π.χ. εκτέλεση προγράμματος, εκτύπωση κώδικα κλπ. είτε με το εργασιακό περιβάλλον π.χ. άνοιγμα / κλείσιμο οθόνης. Δεν θα υπάρχει διαδικασία που δεν θα είναι ρεαλιστική. Τέλος, οι ήχοι, καθώς και τα γραφικά θα είναι όσο το δυνατόν πιο ρεαλιστικά ώστε να νιώθει ο χρήστης ότι βρίσκεται όντως σε υπαρκτό γραφείο προγραμματιστή μιας εταιρείας λογισμικού και όχι σε κάποιο φανταστικό περιβάλλον.

- ✔ **Έλεγχος:** Ο χρήστης θα διαχειρίζεται εργαλεία εργασίας, τα οποία του είναι ιδιαίτερα γνώριμα & θα αντιλαμβάνεται άμεσα τον σκοπό τους. Θα υπάρχει επιλογή γλώσσας, στις οποίες θα περιλαμβάνεται και η Ελληνική.

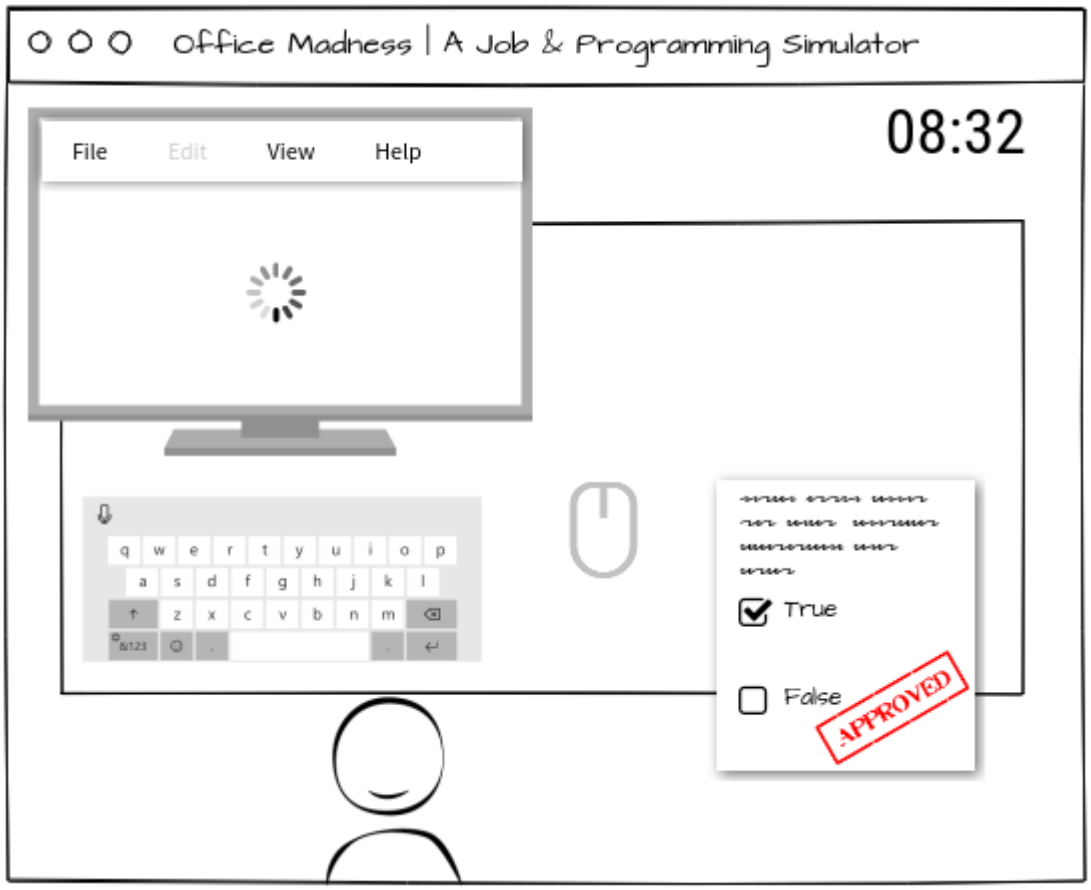
- ✔ **Μεταμόρφωση του χρόνου:** Οι διαδικασίες επίλυσης των εργασιακών προβλημάτων θα είναι συγκεκριμένες & άμεσες. Ο χρήστης – υπάλληλος θα διαβάζει το προγραμματιστικό πρόβλημα που καλείται να επιλύσει, θα το κατανοεί, θα το επιλύει σύμφωνα με τον τύπο προβλήματος π.χ. σύνταξη κώδικα, θα το εκτελεί & τέλος θα το παραδίδει πίσω, σε έντυπη μορφή. Όλες οι διαδικασίες θα απαιτούν συγκεκριμένο & γρήγορο χρόνο επίλυσης, κάτι που θα βοηθάει τον χρήστη να αντιλαμβάνεται καλύτερα την έννοια του χρόνου χρήσης με το παιχνίδι, χωρίς να τον κουράζει.

- ✔ **Αυτόνομη εμπειρία:** Ο χρήστης θα δέχεται, σε μορφή κώδικα, τις προγραμματιστικές αναθέσεις, ωστόσο θα έχει την πλήρη ευθύνη στην επίλυση τους, ανάλογα με τον τύπο του προβλήματος π.χ. σύνταξη κώδικα ή επιλογή κώδικα από μενού. Ο χρήστης θα νιώθει ικανοποιημένος από την επίλυση ενός προβλήματος καθώς θα ακολουθείται ολόκληρη η διαδικασία επίλυσης, από την ανάθεση μέχρι την παράδοση. Κάποιες ενδιάμεσες διαδικασίες όπως π.χ. εκτέλεση κώδικα ή

εκτύπωση του σωστού κώδικα θα προάγουν ακόμη περισσότερο τον ρεαλισμό και εν τέλει την τελική ικανοποίηση του χρήστη. Κάθε ημέρα εργασίας θα παρέχει είτε εκπαίδευση σε νέες προγραμματιστικές έννοιες – π.χ. μέσω κειμένων με παραδείγματα στον Η/Υ – είτε θα χτίζει σε γνώση προηγούμενων εργασιακών ημερών. Τέλος, θα υπάρχει μεγάλο μέγεθος περιεχομένου στο παιχνίδι, που θα καλύπτει αρκετές προγραμματιστικές έννοιες της γλώσσας C++ και θα ικανοποιεί τον σοβαρό χρήστη του παιχνιδιού.

- ✔ **Ενδιαφέρον:** Θα υπάρχουν ελκυστικοί οπτικοί μηχανισμοί παρουσίασης για κάθε περιεχόμενο στο παιχνίδι π.χ. τίτλος παιχνιδιού, εκμάθηση νέων εννοιών, σύνταξη κώδικα, εκτύπωση προγράμματος κλπ. Επίσης η ύπαρξη ρεαλιστικού σεναρίου, οι εργασιακοί στόχοι, το συναφές καθώς και μεγάλο εκπαιδευτικό περιεχόμενο.

- ✔ **Ικανοποίηση:** Ύπαρξη ισορροπημένης & ρεαλιστικής δυσκολίας λόγω των διαβαθμισμένης δυσκολίας στις προγραμματιστικές αναθέσεις με βάση τη θέση εργασίας. Η γνώση που θα παράγεται θα είναι συναφής με τις πραγματικές απαιτήσεις του κόσμου της πληροφορικής.



Εικόνα 8: Office Madness - Mockup οθόνης παιχνιδιού

3.4.6 Τεχνολογική υποδομή

Για την ανάπτυξη της εφαρμογής, χρησιμοποιείται συγκεκριμένο hardware & λειτουργικό σύστημα, καθώς και ένα σύνολο εργαλείων και βιβλιοθηκών (API) που θα αναλυθούν με λεπτομέρεια στο κεφάλαιο της υλοποίησης:

- ✔ **Intel Core i3-5005U CPU @ 2.00 GHz, 64-bit:** Το σύστημα αφορά ένα laptop, χτισμένο για δημιουργία εφαρμογών. Διαθέτει τον γρήγορο επεξεργαστή Core i3 της Intel, 4 GB μνήμης, καθώς και 2 σκληρούς δίσκους των 250 GB & 1 TB αντίστοιχα, με τον πρώτο να είναι SSD και ο οποίος στεγάζει το λειτουργικό σύστημα. Τέλος, στο σύστημα υπάρχουν εγκατεστημένα αρκετά λειτουργικά συστήματα, σε μορφή εικονικών μηχανών, στις οποίες μπορούν να δοκιμαστούν αλλά και να αναπτυχθούν εφαρμογές που προορίζονται να είναι cross-platform π.χ. Android, Linux ή και Mac-OSX.

- ✔ **Windows 7, 64-bit:** Το λειτουργικό σύστημα Windows 7 χρησιμοποιήθηκε για την ανάπτυξη της εφαρμογής. Αποτελεί ένα από τα σταθερότερα & καλά σχεδιασμένα λειτουργικά συστήματα της οικογένειας των Windows.

4 Υλοποίηση του Παιχνιδιού

4.1 Εισαγωγή

Ο κύριος σκοπός της παρούσας εργασίας είναι η ανάπτυξη ενός παιχνιδιού σοβαρού σκοπού με χρήση C++ και OpenGL. Ένας δευτερεύων σκοπός της εργασίας είναι να παρουσιάσει με τρόπο κατανοητό τις διαδικασίες που θα ακολουθηθούν, τόσο σε λογική όσο και σε κώδικα, ώστε να είναι εφικτή η ανάπτυξη ενός παρεμφερούς παιχνιδιού από αναγνώστες της εργασίας.

Η υλοποίηση του παιχνιδιού θα χωριστεί σε διάφορα στάδια ώστε να μπορεί να περιγραφεί ευκολότερα η λογική των διαδικασιών καθώς και να είναι πιο στοχευμένος ο κώδικας που θα παρουσιάζεται. Για λόγους πρακτικούς, δεν θα παρουσιαστεί όλος ο κώδικας του παιχνιδιού αλλά κάποια βασικά τμήματα που καθόρισαν σημαντικά την εφαρμογή καθ' όλη την ανάπτυξη της, τόσο σε τεχνολογικό όσο και σε εκπαιδευτικό επίπεδο.

4.2 Τεχνολογίες

Για την ανάπτυξη της εφαρμογής αυτής χρησιμοποιείται ένα σύνολο εργαλείων και βιβλιοθηκών (API), τα οποία περιγράφονται, αναλυτικά, παρακάτω. Για κάθε εργαλείο ή βιβλιοθήκη, περιγράφεται ο λόγος για τον οποίο απαιτείται η χρήση τους στην εφαρμογή, ενώ αναφέρεται και η *άδεια χρήσης* (license) που παρέχουν. Το τελευταίο είναι πολύ σημαντικό καθώς καθορίζει τι μπορούμε να κάνουμε με τα εργαλεία αυτά και τι έλεγχο έχουμε στη χρήση της εφαρμογής μας από άλλους χρήστες.

4.2.1 Notepad++

Ένας εκ των κορυφαίων κειμενογράφων της αγοράς. Χρησιμοποιήθηκε για να συνταχθεί το σύνολο του κώδικα της εφαρμογής. Δεν χρησιμοποιήθηκε κάποιο συγκεκριμένο IDE όπως το *Visual Studio* ή το *Eclipse*. Ο λόγος που χρησιμοποιείται ο συγκεκριμένος κειμενογράφος αντί κάποιου IDE είναι λόγω ευκολίας & προσβασιμότητας για τον μέσο αναγνώστη, ο οποίος μπορεί να μην έχει ιδιαίτερες γνώσεις προγραμματισμού, αντιλαμβάνεται όμως την χρησιμότητα ενός κειμενογράφου, αλλά και να γίνει επίδειξη της παραγωγής εκτελέσιμων

προγραμμάτων μέσω χρήσης command line & αρχείων script. Η έκδοση της εφαρμογής που χρησιμοποιήθηκε είναι η *v7.2.2* (2016).

Ιστοσελίδα: https://notepad-plus-plus.org

4.2.2 Paint.NET

Πρόκειται για μια δημοφιλή εφαρμογή επεξεργασίας γραφικών για το λειτουργικό σύστημα των *Windows*. Είναι δωρεάν, απλή στη χρήση της και έχει μεγάλη υποστήριξη από προσαρμοσμένα plugins, δημιουργημένα από χρήστες, τα οποία αναβαθμίζουν την εφαρμογή σε πολλές λειτουργίες. Χρησιμοποιήθηκε για την επεξεργασία των γραφικών της εφαρμογής.

Η κύρια εξάρτηση της εφαρμογής είναι το *.NET Framework*, ενός συνόλου τεχνολογιών της Microsoft, το οποίο είναι εγκαταστημένο στα Windows. Ωστόσο, κάποιες εφαρμογές ενδέχεται να απαιτούν αναβαθμισμένη έκδοση. Η έκδοση του *Paint.NET* που χρησιμοποιήθηκε είναι η *v4.2.5* (2019).

Ιστοσελίδα: https://www.getpaint.net

4.2.3 MinGW-w64

Αλλιώς και *Minimalist GNU for Windows 32 & 64 bits*, είναι ένας δωρεάν compiler για C++, ο οποίος χρησιμοποιήθηκε για την παραγωγή των εκτελέσιμων αρχείων της εφαρμογής. Πρόκειται για μια επέκταση του compiler *MingGW*, με υποστήριξη ωστόσο και για αρχιτεκτονικές 64-bit.

Σε αντίθεση με άλλους compilers – όπως αυτός του *Visual Studio* της Microsoft ή το *Eclipse* – είναι ότι δεν διαθέτει ενσωματωμένο IDE, λειτουργεί δηλαδή μόνο σε γραμμή εντολών (command line).

Δεν έχει εξαρτήσεις από άλλες βιβλιοθήκες, εκτός από κάποιες βασικές βιβλιοθήκες του λειτουργικού συστήματος των Windows, είναι cross-platform ενώ η άδεια (license) του compiler έχει οριστεί σε public domain, δεν υπάρχει δηλαδή κανένας περιορισμός με την χρήση του και έτσι τα παραγόμενα εκτελέσιμα αρχεία μπορούν να χρησιμοποιούν για οποιοδήποτε σκοπό. Η έκδοση του compiler που χρησιμοποιήθηκε είναι η *v8.1.0* (2018).

Ιστοσελίδα: http://mingw-w64.org/doku.php

4.2.4 Digital Mars C/C++ Compiler

Η εφαρμογή επιτρέπει στους χρήστες να πληκτρολογούν κώδικα σε C++, οπότε με κάποιον τρόπο πρέπει να επαληθεύεται η ορθότητα τους. Αυτό πραγματοποιείται μέσω του compiler C/C++ της *Digital Mars*, μιας εταιρείας που ειδικεύεται σε compilers *μικρού μεγέθους* και οι οποίοι εκτελούνται στην γραμμή εντολών.

Η επιλογή του συγκεκριμένου compiler ήταν αρκετά εύκολη, αν ληφθούν οι απαιτήσεις & ιδιαιτερότητες της εφαρμογής. Αρχικά, έπρεπε ο compiler να έχει *μικρό μέγεθος* σε απαιτήσεις δίσκου. Η γλώσσα προγραμματισμού C++ είναι αρκετά περίπλοκη, με αποτέλεσμα οι υλοποιήσεις μεταγλωττιστών να έχουν απαγορευτικό μέγεθος ώστε να μπορούν να μεταφερθούν σε μια εφαρμογή & να διανέμονται σε χρήστες.

Για παράδειγμα, ο μεταγλωττιστής *MinGW-w64*, ο οποίος χρησιμοποιείται για να μεταγλωττιστεί το σύνολο της εφαρμογής και υποστηρίζει μόνο γραμμή εντολών, απαιτεί χώρο περίπου *200mb*, μέγεθος απαγορευτικό για μια εφαρμογή της οποίας το συνολικό μέγεθος κυμαίνεται στα *~25mb* περίπου. Με την συμπερίληψη και του *Digital Mars* compiler, το συνολικό μέγεθος της εφαρμογής δεν ξεπερνάει τα *50mb*, μέγεθος αρκετά λογικό που μειώνεται και με συμπίεση.

Μια δεύτερη απαίτηση ήταν να καλύπτεται τουλάχιστον η έκδοση *C++98*. Η έκδοση του compiler που χρησιμοποιείται καλύπτει μέχρι και *C++11* (2011), το οποίο σημαίνει ότι μπορεί να μεταγλωττίσει & να επαληθεύσει την ορθότητα κώδικα σε όλες τις ασκήσεις προγραμματισμού της εφαρμογής, των οποίων ο κώδικας συντάχθηκε ώστε να είναι συμβατός με την έκδοση *C++11*. Παράλληλα, υποστηρίζει πλήρως την βιβλιοθήκη STL της C++, επιτρέποντας την χρήση *vectors, maps, iterators* και πολλές ακόμα λειτουργίες της C++.

Το συνολικό μέγεθος του compiler της *Digital Mars* είναι περίπου *25mb* και περιλαμβάνεται σε ένα συγκεκριμένο folder της εφαρμογής. Κατά την εκτέλεση της, η εφαρμογή εκτελεί τόσο τον compiler όσο και τον linker ώστε να μεταγλωττίζει, σε πραγματικό χρόνο, τον κώδικα που γράφουν οι χρήστες & να επαληθεύει την ορθότητα τους, όπως ακριβώς θα γινόταν και σε ένα IDE, ωστόσο μόνο με τη χρήση του ενσωματωμένου συντάκτη κώδικα της εφαρμογής. Με αυτό τον τρόπο οι χρήστες έχουν όλη την ανάδραση που απαιτείται ώστε να καταλάβουν τα λάθη τους & να εκπαιδευτούν στη σύνταξη

κώδικα. Η έκδοση του μεταγλωττιστή που χρησιμοποιήθηκε είναι η *v8.57* (2013) και αποτελεί την πιο ενημερωμένη έκδοση.

Ιστοσελίδα: https://digitalmars.com/download/freecompiler.html

4.2.5 Win32 API

Είναι το API στο οποίο στηρίζεται όλη η λειτουργία των Windows. Χρησιμοποιείται μόνο στα Windows και έτσι δεν απαιτείται σε μια εφαρμογή που πρόκειται να εκτελεστεί π.χ. στο Linux ή στο Android. Είναι γραμμένο στην γλώσσα προγραμματισμού C και έτσι παρέχει ένα διαδικαστικό API.

Οι λειτουργίες που παρέχει είναι χαμηλού επιπέδου π.χ. δημιουργία παραθύρων, γραφική διασύνδεση παραθύρων, διαχείριση εισόδου / εξόδου, διαχείριση μνήμης κ.α. Κάθε έκδοση του API βασίζεται στο παρόν λειτουργικό σύστημα και (συνήθως) είναι συμβατό με παλιότερες εκδόσεις των Windows.

Τέλος, η γνώση της απαιτούμενης άδειας χρήσης (license) δεν εξυπηρετεί ιδιαίτερα λόγω ότι το Windows API είναι πάντα διαθέσιμο σε κάθε υπολογιστικό σύστημα που τρέχει Windows.

Ιστοσελίδα: https://docs.microsoft.com/en-us/windows/win32/api

4.2.6 OpenGL

Είναι το cross-platform API που χρησιμοποιήθηκε για την αναπαραγωγή των γραφικών του παιχνιδιού. Αποτελεί ένα σταθερό & εξελιγμένο API γραφικών και το σύνολο των κατασκευαστών καρτών γραφικών (GPU), όπως επίσης και πολλές μηχανές γραφικών, το υποστηρίζουν.

Κατά την ανάπτυξη της εφαρμογής, αξιοποιήθηκε η έκδοση *2.0* της OpenGL (2004), η οποία ήταν και η πρώτη έκδοση που υποστήριζε το *GLSL*, την κύρια γλώσσα προγραμματισμού των καρτών γραφικών για OpenGL.

Ο λόγος που έγινε χρήση της πρώιμης αυτής έκδοσης αφορά κυρίως λόγους συμβατότητας με τις συσκευές στις οποίες ενδεχομένως θα εκτελεστεί η εφαρμογή αλλά και για λόγους απλότητας των αρχικών εκδόσεων έναντι των νεότερων που υποστηρίζουν συνεχώς νέα (αλλά και περίπλοκα) πρότυπα προγραμματισμού.

Τέλος, η γνώση της απαιτούμενης άδειας χρήσης (license) δεν εξυπηρετεί ιδιαίτερα λόγω του ότι το OpenGL API είναι σχεδόν πάντα διαθέσιμο και υλοποιημένο στους οδηγούς (drivers) των καρτών γραφικών.

Ιστοσελίδα: https://www.opengl.org

4.2.7 GLEW

Είναι τα αρχικά του *OpenGL Extension Wrangler* και αφορά μια cross-platform βιβλιοθήκη γραμμένη σε C/C++, σκοπός της οποίας είναι ουσιαστικά να κάνει ευκολότερη τη διασύνδεση μεταξύ OpenGL και εφαρμογής. Όπως έχει αναφερθεί, το OpenGL αφορά μια δυναμική βιβλιοθήκη, υλοποιημένη σε μορφή driver από τον κατασκευαστή της κάρτας γραφικών.

Ωστόσο, εάν η κάρτα γραφικών δεν υποστηρίζει όλα τα χαρακτηριστικά του OpenGL, δεν θα υλοποιεί και την αντίστοιχη λειτουργικότητα στον driver που παρέχει. Εάν η εφαρμογή θέλει να χρησιμοποιήσει μια λειτουργία του OpenGL, η οποία ωστόσο δεν είναι διαθέσιμη στον driver, θα προκαλέσει πρόβλημα και κλείσιμο της εφαρμογής κατά την εκκίνηση της.

Ένας τρόπος να λυθεί το συγκεκριμένο πρόβλημα είναι να αντιληφθεί, δυναμικά, η εφαρμογή τις διαθέσιμες δυνατότητες του OpenGL driver και να προσαρμόσει την λειτουργική συμπεριφορά της, αποφεύγοντας την χρήση λειτουργιών που δεν είναι διαθέσιμες. Τη λειτουργία αυτή αναλαμβάνει η βιβλιοθήκη *GLEW*, διαβάζοντας & παρέχοντας στην εφαρμογή το σύνολο των διαθέσιμων λειτουργιών του OpenGL driver, έτοιμων προς χρήση.

Η έκδοση που χρησιμοποιήθηκε είναι η *v2.0.0* (2016). Τέλος, η άδεια χρήσης (license) που παρέχει είναι μια μίξη *BSD / MIT* (του ομώνυμου πανεπιστημίου), οι οποίες είναι αρκετά ευέλικτες και μας επιτρέπουν τη χρήση της βιβλιοθήκης ουσιαστικά για κάθε σκοπό.

4.2.8 OpenAL Soft

Είναι η software υλοποίηση του γνωστού cross-platform API για 3D ηχητική αναπαραγωγή OpenAL. Υποστηρίζει τις περισσότερες hardware λειτουργίες του OpenAL, ωστόσο λόγω του ανοιχτού κώδικα μπορεί να

υποστηριχθεί και σε άλλα συστήματα. Το όνομα επιλέχθηκε ώστε να θυμίζει το OpenGL, καθώς το API του έχει παραπλήσια μορφή σύνταξης.

Η έκδοση που χρησιμοποιήθηκε είναι η *v1.17.2* (2016). Τέλος, η άδεια χρήσης (license) που παρέχει είναι η *LGPL*, αλλιώς και GNU Lesser General Public License. Λόγω της άδειας αυτής, η εφαρμογή θα κάνει χρήση της βιβλιοθήκης σε *δυναμική* μορφή *(ξεχωριστό αρχείο .dll)*.

Ιστοσελίδα: https://openal-soft.org

4.2.9 PugiXML

Αφορά μια C++ βιβλιοθήκη, με την οποία διαχειριζόμαστε XML αρχεία με οργανωμένο τρόπο. Το εκπαιδευτικό περιεχόμενο είναι αποθηκευμένο σε αρχεία XML με οργανωμένο τρόπο. Η αυτοματοποιημένη αναζήτηση της κατάλληλης πληροφορίας στα αρχεία XML από μια εφαρμογή, προϋποθέτει κάποια αρχικά δεδομένα και υλοποιημένο μηχανισμό, τον οποίο παρέχει η βιβλιοθήκη που χρησιμοποιούμε.

Η έκδοση που χρησιμοποιήθηκε είναι η *v1.5* (2014). Τέλος, η άδεια χρήσης (license) που παρέχει η βιβλιοθήκη είναι η *MIT* (του ομώνυμου πανεπιστημίου), η οποία είναι αρκετά ευέλικτη και μας επιτρέπει την χρήση της βιβλιοθήκης ουσιαστικά για κάθε σκοπό.

Ιστοσελίδα: https://pugixml.org

4.2.10 FreeType

Είναι μια δημοφιλής βιβλιοθήκη, γραμμένη στη γλώσσα προγραμματισμού C, μέσω της οποίας γίνεται διαχείριση αρχείων γραμματοσειρών. Η εφαρμογή χρησιμοποιεί κείμενο για να μεταφέρει την εκπαιδευτική πληροφορία στον χρήστη και σε *τουλάχιστον* δυο (2) γλώσσες. Τα αρχεία γραμματοσειρών (με καταλήξεις *.ttf* και *.otf*) καθορίζουν και τις γλώσσες που υποστηρίζονται.

Η έκδοση που χρησιμοποιήθηκε είναι η *v2.5.3* (2014). Τέλος, η άδεια χρήσης (license) που παρέχει η βιβλιοθήκη είναι η (ομώνυμη) *FreeType License*, η οποία είναι αρκετά ευέλικτη και επιτρέπει τη χρήση της βιβλιοθήκης ουσιαστικά για κάθε σκοπό.

Ιστοσελίδα: https://www.freetype.org

4.3 Μηχανή παιχνιδιού

Όπως αναφέρθηκε στην εισαγωγή της εργασίας, η εφαρμογή έχει αναπτυχθεί χωρίς τη χρήση μιας έτοιμης μηχανής παιχνιδιών όπως η Unity. Ωστόσο, η οργανωμένη & σωστή δημιουργία ενός παιχνιδιού ή μιας εφαρμογής γενικότερα προϋποθέτει ένα API, επάνω στο οποίο θα στηριχθούν όλες οι απαιτούμενες λειτουργίες.

Έτσι, δημιουργήθηκε ένα API το οποίο συγκέντρωσε όλες τις κλάσεις, μεθόδους & δεδομένα της εφαρμογής, παίρνοντας σιγά-σιγά την μορφή μιας *προσαρμοσμένης μηχανής παιχνιδιών*, στοχευμένης μόνο στις ανάγκες της εφαρμογής και χωρίς να περιλαμβάνει δυνατότητες, οι οποίες ενώ θα ήταν βασικές σε άλλες μηχανές παιχνιδιών, δεν χρησιμοποιήθηκαν στην εφαρμογή.

4.3.1 Διαφορές με μηχανές παιχνιδιών γενικής χρήσης

Οι διαφορές, ωστόσο, με μηχανές παιχνιδιών γενικής χρήσης είναι αρκετές, με τις κυριότερες να παρουσιάζονται παρακάτω:

- ✔ **Εκμάθηση χρήσης:** Η προσαρμοσμένη μηχανή παιχνιδιών αφορά λογισμικό το οποίο αναπτύχθηκε παράλληλα με την ανάπτυξη της κύριας εφαρμογής, η οποία και θα την χρησιμοποιήσει. Με αυτόν τον τρόπο *προσπεράστηκε,* με έμμεσο τρόπο, η διαδικασία εκμάθησης της χρήσης μηχανής παιχνιδιών.

- ✔ **Προγραμματιστική βελτίωση:** Οι προγραμματιστές που επενδύουν χρόνο στη δημιουργία κάποιας προσαρμοσμένης μηχανής παιχνιδιών, αντιμετωπίζουν συνέχεια προγραμματιστικές προκλήσεις σε χαμηλό επίπεδο. Η συνεχής έρευνα που αποσκοπεί στην επίλυση των προκλήσεων αυτών ενισχύει την προγραμματιστική γνώση που έχουν, ωστόσο αυτό συνήθως δεν προσθέτει κάτι στην τελική εφαρμογή.

- ✔ **Χρόνος:** Η ανάπτυξη προσαρμοσμένης μηχανής παιχνιδιών, παράλληλα με την ανάπτυξη μιας εφαρμογής η οποία θα την χρησιμοποιεί σημαίνει ανάπτυξη δυο (2) εφαρμογών, εκ των οποίων η μια (συνήθως) θα διατεθεί

στο κοινό. Αυτό σημαίνει μεγαλύτερη επένδυση χρόνου και συνήθως δεν αντανακλάται στο τελικό προϊόν που παραδίδεται στον χρήστη.

- ✔ **Έλεγχος:** Ο κώδικας της προσαρμοσμένης μηχανής παιχνιδιών είναι διαθέσιμος και μπορεί να τροποποιηθεί ή και να βελτιστοποιηθεί με όποιον τρόπο αυτό απαιτηθεί από την εφαρμογή. Κάτι τέτοιο συνήθως είναι αρκετά δύσκολο & χρονοβόρο, μέχρι και ανέφικτο (όταν δεν υπάρχει διαθέσιμος ο κώδικας) με μηχανές παιχνιδιών γενικής χρήσης.

- ✔ **Προσαρμογή:** Ο κώδικας της προσαρμοσμένης μηχανής παιχνιδιών είναι αυτό ακριβώς, προσαρμοσμένος στις απαιτήσεις μιας εφαρμογής. Μια μηχανή παιχνιδιών γενικής χρήσης απαιτεί την υλοποίηση πολλών μηχανισμών από πριν επειδή ακριβώς δεν μπορεί να γίνει πρόβλεψη των διαφόρων τύπων εφαρμογών που θα παράξει, των χρηστών που θα την χρησιμοποιήσουν, της αρχιτεκτονικής στην οποία θα χρησιμοποιηθεί κ.α.

Η προσαρμοσμένη μηχανή παιχνιδιών ονομάστηκε **JoIt3D.** Το API της μηχανής παιχνιδιών αποφασίστηκε να είναι διαδικαστικό για λόγους απλότητας. Οι κλήσεις προς την μηχανή παιχνιδιών έχουν το πρόθεμα `J3D_xxx` ώστε να ξεχωρίζουν από τις κλήσεις προς τα άλλα APIs της εφαρμογής.

4.3.2 Κύριες λειτουργίες

Η προσαρμοσμένη μηχανή παιχνιδιών υποστηρίζει αρκετές λειτουργίες, οι κυριότερες των οποίων περιγράφονται παρακάτω:

- ✔ **Cross platform:** Η προσαρμοσμένη μηχανή παιχνιδιών μπορεί να εκτελέσει τις περισσότερες λειτουργίες της σε ένα σύνολο λειτουργικών συστημάτων, συγκεκριμένα στα λειτουργικά συστήματα *Windows, Linux, Android, MacOSX, iOS* και *Solaris*. Ωστόσο, στην εργασία γίνεται συνδυασμός των διαφορετικών API των λειτουργικών συστημάτων κατά την παρουσίαση τμημάτων κώδικα, για μεγαλύτερη ολοκλήρωση.

- ✔ **Callbacks:** Αφορά τμήματα κώδικα τα οποία εκτελούνται σε συγκεκριμένες στιγμές, με βάση την λειτουργικότητα τους. Για παράδειγμα, το τμήμα κώδικα που σχεδιάζει στην οθόνη και το τμήμα κώδικα που φορτώνει αρχικά δεδομένα και αρχικοποιεί την μηχανή θα πρέπει να εκτελεστούν σε διαφορετικές στιγμές.

- ✔ **Αυτόματη ρύθμιση της ανάλυσης:** Η εφαρμογή έχει αναπτυχθεί σε συγκεκριμένη ανάλυση. Ωστόσο, οι περισσότερες αναλύσεις οθόνης των χρηστών της εφαρμογής θα διαφέρουν. Η μηχανή παιχνιδιών προσαρμόζεται *αυτόματα* στην τρέχουσα ανάλυση οθόνης του χρήστη.

- ✔ **Δείκτης ποντικιού:** Αφορά την φόρτωση & ορισμό δεικτών ποντικιού στην οθόνη. Οι δείκτες ποντικιών (ή κέρσορες) βρίσκονται σε κάποια γραφική μορφή στον σκληρό δίσκο (π.χ. αρχεία *.cur* ή *.png*). Η μηχανή παιχνιδιών μπορεί να τα φορτώσει στη μνήμη και να τα εμφανίσει στην οθόνη, παραδίδοντας έπειτα τον έλεγχο στο ποντίκι του χρήστη.

- ✔ **Shaders:** Αφορά την φόρτωση & διαχείριση των shaders που εκτελούνται στην κάρτα γραφικών. Η ίδια η 2D λειτουργικότητα που παρέχεται από την προσαρμοσμένη μηχανή παιχνιδιών δημιουργείται δυναμικά ως shader, όπως επίσης και η αυτόματη ρύθμιση της ανάλυσης οθόνης.

- ✔ **Γραμματοσειρές:** Η προσαρμοσμένη μηχανή παιχνιδιών κάνει χρήση της δημοφιλούς βιβλιοθήκης γραμματοσειρών **FreeType,** παρέχοντας φόρτωση πολυγλωσσικών & ανεξάρτητων του λειτουργικού συστήματος αρχείων γραμματοσειρών όπως *.ttf* (truetype), *.otf* (opentype) και άλλων.

- ✔ **Textures:** Αφορά γραφικά τα οποία είναι αποθηκευμένα στην μνήμη της κάρτας γραφικών. Χρησιμοποιούνται τόσο σε 2D όσο και 3D π.χ. κάλυψη ενός τρισδιάστατου μοντέλου με γραφικά. Ωστόσο στην προσαρμοσμένη μηχανή παιχνιδιών χρησιμοποιούνται για την γραφική αναπαράσταση όλων των 2D γραφικών που υπάρχουν στο παιχνίδι π.χ. γραφείο, οθόνη, πληκτρολόγιο, γραμματοσειρές, ψηφιακό ρολόϊ κ.α. Η προσαρμοσμένη μηχανή παιχνιδιών υποστηρίζει την φόρτωση εικόνων από τον σκληρό

δίσκο απευθείας στην κάρτα γραφικών, διαχείριση σε επίπεδο pixels, άμεση σχεδίαση γραφικών σε άλλα textures (render to texture) αντί της οθόνης, καθώς και άλλες δυνατότητες.

- ✔ **Ήχος:** Η προσαρμοσμένη μηχανή παιχνιδιών χρησιμοποιεί το δημοφιλές cross-platform API διαχείρισης ήχου OpenAL για την αναπαραγωγή αρχείων ήχου. Υποστηρίζεται η φόρτωση συμπιεσμένων αρχείων ήχου *.ogg* (ogg vorbis), τα οποία είναι παραπλήσια των *.mp3* αρχείων, ωστόσο είναι βασισμένα στην ευέλικτη άδεια *BSD* και έτσι η χρήση τους δεν υπόκειται σε περιορισμούς που ενδεχομένως να υπάρχουν για την χρήση αρχείων *.mp3*. Η μηχανή παιχνιδιών υποστηρίζει άμεση φόρτωση των αρχείων ήχου στη μνήμη ή τμηματικής όσο παίζουν *(streaming)*, επιτρέποντας καλύτερη διαχείριση μνήμης.

- ✔ **2D γραφικά:** Η προσαρμοσμένη μηχανή παιχνιδιών υποστηρίζει μόνο 2D γραφικά λόγω της αισθητικής του παιχνιδιού και έτσι δεν υπάρχει 3D υποστήριξη καθώς δεν χρησιμοποιείται καθόλου από την εφαρμογή. Οι κύριες λειτουργίες 2D αφορούν προσαρμοσμένη σχεδίαση κειμένου στην οθόνη, προσαρμοσμένη σχεδίαση εικόνων & γραφικών που έχουν φορτωθεί από τον σκληρό δίσκο, σχεδίαση πολυγωνικών & ελλειψοειδών σχημάτων κ.α. Η προσαρμοσμένη σχεδίαση αφορά λειτουργίες κατά τις οποίες τα αντικείμενα γραφικών μπορούν να αυξομειώσουν το μέγεθος ή το χρώμα τους, να τους αλλάξουν κατεύθυνση, να τα περιστρέψουν κ.α.

- ✔ **Χρονικό animation:** Παρέχει λειτουργίες κατά τις οποίες δημιουργούνται χρονομετρητές με διάφορες παραμέτρους, οι οποίοι χρησιμοποιούνται για εκτέλεση προγραμματισμένων συμβάντων όσο εκτελείται η εφαρμογή. Με τη λειτουργικότητα αυτή προγραμματίζονται στην εφαρμογή γραφικά animations π.χ. άνοιγμα οθόνης & εκτύπωση κώδικα, αλλά και συμβάντα μεγάλης χρονικής κάλυψης π.χ. ψηφιακό ρολόι κ.α.

- ✔ **Έλεγχος:** Αφορά την διαχείριση ελέγχου των συσκευών εισόδου του χρήση, όπως ποντίκι & πληκτρολόγιο. Η εφαρμογή αντιδράει στις ενέργειες χειρισμού του χρήστη.

- ✔ **Διαχείριση αρχείων:** Παρέχει λειτουργικότητα για την επεξεργασία αρχείων στον σκληρό δίσκο π.χ. έλεγχος ύπαρξης αρχείων & καταλόγων, γράψιμο σε αρχεία & ανάγνωση από αυτά, αντιγραφή, μεταφορά & διαγραφή αρχείων, δημιουργία & διαχείριση συμπιεσμένων αρχείων κ.α.

4.4 Δομή εφαρμογής

Οι εφαρμογές οι οποίες είναι γραμμένες σε C++, απαιτούν την ύπαρξη μιας αρχικής συνάρτησης με όνομα `main()`. Η συνάρτηση αυτή είναι το entry point κάθε εφαρμογής C++, είναι το σημείο δηλαδή από το οποίο ξεκινάει η εκτέλεση των εντολών ενός προγράμματος, παρέχοντας (προαιρετικά) τις παραμέτρους της εφαρμογής.

4.4.1 Απλή δομή εφαρμογής

Μια τυπική δομή εφαρμογής στη γλώσσα C++, η οποία ωστόσο δεν υλοποιεί καμία απολύτως λειτουργικότητα, μπορεί όμως να μεταγλωττιστεί με επιτυχία σε εκτελέσιμο πρόγραμμα και να θεωρηθεί ολοκληρωμένη εφαρμογή, παρουσιάζεται στον Κώδικα 1:

Κώδικας 1: Δομή εφαρμογής στη γλώσσα C++

```
int main(int argc, char** argv) {
    return 0;
}
```

Η παραπάνω μορφή, όπως αναφέρθηκε, δεν υλοποιεί καμία απολύτως λειτουργικότητα. Είναι όμως απαραίτητη η ύπαρξη μιας συνάρτησης `main()`, καθώς και η επιστροφή μιας τιμής της προς το λειτουργικό σύστημα.

4.4.2 Δομή εφαρμογής με προσαρμοσμένη μηχανή παιχνιδιών

Σε μια περίπλοκη εφαρμογή όπως ένα βιντεοπαιχνίδι ή μια εφαρμογή σοβαρού σκοπού, η δομή της εφαρμογής απαιτεί περισσότερο κώδικα & κλήσεις συναρτήσεων ώστε να αρχικοποιηθεί, να δημιουργήσει τα κατάλληλα αντικείμενα, το παράθυρο της εφαρμογής & τέλος να δώσει τον έλεγχο στον

χρήστη. Στον Κώδικα 2, παρουσιάζεται η δομή εφαρμογής που έχει υλοποιηθεί για τη χρήση μέσω της προσαρμοσμένης μηχανής παιχνιδιών:

Κώδικας 2: Δομή εφαρμογής με προσαρμοσμένη μηχανή παιχνιδιών

```
#include "../JOLT3D/jolt3d.h"
#include "pugixml/pugixml.hpp"
#include "game.cpp"
#include "title.cpp"
...

void J3D_callback_window(J3D_WINDOWEVENT e) {
    switch(e.event) {
        case J3D_WINDOW_INIT:
            ...
        break;
        case J3D_WINDOW_DEINIT:
        break;
        case J3D_WINDOW_LOOP:
            ...
        break;
    }
}

void J3D_main() {
    J3D_init();
    J3D_display_set_resolution(1024, 768, true);
    J3D_window_create("main", "", 50, 50, 400, 300, "The Game v1.0",
                      false);
    J3D_loop();
}
```

Η δομή αυτή έχει συγκεκριμένη μορφή, βασισμένη στις απαιτήσεις της προσαρμοσμένης μηχανής παιχνιδιού. Κάθε μηχανή παιχνιδιών έχει την δική της δομή, όπως επίσης και τα δικά της αρχεία όπως π.χ. το `/jolt3d.h` το οποίο καλείται στην πρώτη γραμμή. Υπάρχουν τρία (3) διακριτά τμήματα κώδικα της προσαρμοσμένης μηχανής παιχνιδιών, τα οποία είναι:

- ✔ **Αρχεία κεφαλίδας (header):** Τα αρχεία αυτά περιλαμβάνουν τις κλάσεις & μεθόδους άλλων βιβλιοθηκών που χρησιμοποιούνται στην εφαρμογή. Παραδείγματα είναι το API της προσαρμοσμένης μηχανής παιχνιδιών, τα αρχεία πηγαίου κώδικα του παιχνιδιού κ.α.

- ✔ **Συναρτήσεις callback:** Η προσαρμοσμένη μηχανή παιχνιδιών ορίζει στην αρχή κάποιες συναρτήσεις ως *callback* συναρτήσεις. Οι συναρτήσεις αυτές καλούνται κατά την εκτέλεση της εφαρμογής σε συγκεκριμένες στιγμές, ώστε να επιτελούν συγκεκριμένες εργασίες. Στην παραπάνω δομή εφαρμογής εμφανίζεται μια τέτοια callback συνάρτηση με όνομα `J3D_callback_window`, ωστόσο υπάρχουν και άλλες συναρτήσεις callback. Όπως γίνεται αντιληπτό από το όνομα, αφορά μια συνάρτηση callback που εκτελεί εργασίες στο *παράθυρο της εφαρμογής* (window).

- ✔ **Συνάρτηση main:** Η συνάρτηση `main()` και ο σκοπός που επιτελεί σε μια εφαρμογή C++ αναφέρθηκε παραπάνω. Ωστόσο, στην προσαρμοσμένη μηχανή παιχνιδιών έχει μετονομαστεί σε `J3D_main()`. Ο λόγος δεν είναι απλά για να συμβαδίζει με το API της προσαρμοσμένης μηχανής παιχνιδιών. Ο κυριότερος λόγος είναι η δυνατότητα της μηχανής παιχνιδιών να εκτελείται και σε άλλα λειτουργικά συστήματα, κάποια εκ των οποίων δεν χρησιμοποιούν την `main()` ως σημείο εκκίνησης. Για παράδειγμα, στο λειτουργικό σύστημα Android υπάρχει η συνάρτηση `android_main()`. Η βασική ιδέα είναι να γίνεται *εσωτερική* μετονομασία της συνάρτησης `main()` κάθε λειτουργικού συστήματος σε ένα άλλο όνομα, συγκεκριμένα το `J3D_main()`. Με αυτόν τον τρόπο, ο προγραμματιστής χρησιμοποιεί την ίδια βάση κώδικα, ανεξάρτητα του λειτουργικού συστήματος στο οποίο βρίσκεται.

4.5 Μεταγλώττιση εφαρμογής

Για να εκτελεστεί μια εφαρμογή σε γλώσσα C++, θα πρέπει ο πηγαίος κώδικας (αρχεία με επεκτάσεις *.cpp / .h*) να μεταγλωττιστεί σε εκτελέσιμο κώδικα (αρχείο με επέκταση *.exe*). Η διαδικασία αυτή ονομάζεται μεταγλώττιση ή compilation, ενώ τα εργαλεία που την αναλαμβάνουν ονομάζονται *compilers*.

4.5.1 Τρόποι μεταγλώττισης

Υπάρχουν δυο (2) κύριοι τρόποι να γίνει μεταγλώττιση σε μια εφαρμογή:

- ✔ **Με χρήση IDE:** Μέσω IDE, παρέχονται όλα τα απαραίτητα εργαλεία σύνταξης, αποσφαλμάτωσης & μεταγλώττισης κώδικα, κάνοντας τη διαδικασία της ανάπτυξης ευκολότερη. Ωστόσο, η σωστή χρήση ενός IDE προϋποθέτει σημαντικό χρόνο εκμάθησης. Οι παράμετροι της μεταγλώττισης, για παράδειγμα αρχιτεκτονική, βελτιστοποίηση, μονοπάτια βιβλιοθηκών & αρχεία κεφαλίδων κλπ. παρέχονται από το IDE.

- ✔ **Με χρήση αρχείων script:** Μέσω αρχείων script, μπορούν να γραφούν εντολές οι οποίες θα εκτελούν – αυτοματοποιημένα – άλλα προγράμματα, όπως για παράδειγμα τον μεταγλωττιστή της γλώσσας C++, με απώτερο σκοπό να μεταγλωττίσουν πηγαία αρχεία σε εκτελέσιμα αρχεία. Ο τρόπος αυτός είναι σχετικά πιο περίπλοκος, ωστόσο παρέχει μεγαλύτερη ευελιξία & συμβατότητα μεταξύ λειτουργικών συστημάτων. Οι παράμετροι της μεταγλώττισης εδώ παρέχονται με τη μορφή παραμέτρων εκτέλεσης στο εκτελέσιμο πρόγραμμα του μεταγλωττιστή μέσα στο script αρχείο.

4.5.2 Μεταγλώττιση μέσω αρχείου script

Στην εφαρμογή που αναπτύχθηκε, επιλέχθηκε ο δεύτερος τρόπος μεταγλώττισης, συγκεκριμένα με τη χρήση αρχείων script. Αυτό σημαίνει ότι θα χρησιμοποιείται ένα αρχείο script για να γίνεται μεταγλώττιση των πηγαίων αρχείων κώδικα σε εκτελέσιμη μορφή.

Το αρχείο που χρησιμοποιείται για τη μεταγλώττιση της εφαρμογής ονομάστηκε *windows_mingw64_compile.bat*. Η ονοματολογία αυτή του αρχείου

επιλέχθηκε ώστε να καθορίζει επαρκώς τον σκοπό του script αρχείου, συγκεκριμένα αφορά μεταγλώττιση σε σύστημα *Windows* με μεταγλωττιστή τον *MingGW-w64*.

Για παράδειγμα, ένα αρχείο με όνομα *windows_msvs_compile.bat* θα πραγματοποιούσε την ίδια μεταγλώττιση, ωστόσο με χρήση του *Visual Studio* compiler. Με αυτόν τον τρόπο, μπορούμε εύκολα να οργανώσουμε το είδος της μεταγλώττισης που θέλουμε, κατά λειτουργικό σύστημα & compiler.

4.5.3 Αρχείο μεταγλώττισης

Στον Κώδικα 3, εμφανίζεται το περιεχόμενο του αρχείου script, μέσω του οποίου γίνεται η μεταγλώττιση της εφαρμογής σε εκτελέσιμο πρόγραμμα:

Κώδικας 3: Αρχείο script για μεταγλώττιση εφαρμογής

```
@echo off

SET MINGW64_FOLDER=D:\PROGRAMS\mingw-w64\i686-8.1.0-posix-dwarf-rt_v6-rev0\mingw32\bin
SET PATH=%MINGW64_FOLDER%;%PATH%
SET GPP=%MINGW64_FOLDER%\i686-w64-mingw32-g++
SET BINFOLDER="Release/"
SET BINPATH=%BINFOLDER%start.exe
IF NOT EXIST %BINFOLDER% MKDIR %BINFOLDER%

%GPP% -DNDEBUG -s -static -std=c++0x -static-libgcc -static-libstdc++ -O0 main.cpp -L
../JOLT3D/libs_static -L ../JOLT3D/libs_dynamic/windows -Wl,-Bstatic -lJOLT3D_windows
-lstdc++ -lpthread -Wl,-Bdynamic -lwinhttp -lgdi32 -lopengl32 -lopenal32 -Wl,-
subsystem,windows -o %BINPATH%
```

Ο παραπάνω κώδικας εκτελεί *εντολές* προς το λειτουργικό σύστημα, καθώς και το εκτελέσιμο αρχείο του *μεταγλωττιστή* με έναν συγκεκριμένο αριθμό παραμέτρων, με την πιο σημαντική να είναι το μονοπάτι του μεταγλωττιστή *(MINGW64_FOLDER)*, αλλά και το όνομα του πηγαίου αρχείου (*main.cpp)*, όπου υλοποιείται η εφαρμογή. Συγκεκριμένα, τα βήματα που ακολουθούνται στο αρχείο script για να μεταγλωττιστεί η εφαρμογή, είναι τα παρακάτω:

- ✔ *Απενεργοποιεί* την εμφάνιση των εντολών τις οποίες εκτελεί (*@echo off*)
- ✔ Ορίζει κάποιες μεταβλητές (*SET*) για όσο διαρκεί η μεταγλώττιση
- ✔ Ορίζεται το μονοπάτι των αρχείων του μεταγλωττιστή (*MING64_FOLDER*)

- ✔ Ορίζεται το μονοπάτι αυτό στο *PATH* ώστε να είναι παντού προσβάσιμο
- ✔ Ορίζεται το μονοπάτι στο εκτελέσιμο αρχείο του compiler (*GPP*)
- ✔ Ορίζονται τα μονοπάτια του folder & εκτελέσιμου αρχείου της εφαρμογής (*BINFOLDER, BINPATH*)
- ✔ Ελέγχει εάν το folder του εκτελέσιμου αρχείου υπάρχει ή το δημιουργεί
- ✔ Εκτελεί το αρχείο του compiler με έναν αριθμό παραμέτρων
- ✔ Στο τέλος της εκτέλεσης, θα υπάρχει εκτελέσιμο αρχείο στο *BINFOLDER*

Η μεταβλητή *PATH* είναι καθολική μεταβλητή του συστήματος και περιέχει όλα τα μονοπάτια του συστήματος αρχείων, τα αρχεία των οποίων θα είναι προσβάσιμα για εκτέλεση μέσω γραμμής εντολών (άρα και μέσα στο script), χωρίς να απαιτείται το πλήρες μονοπάτι τους. Με την (προσωρινή) τροποποίηση της μεταβλητής *PATH* στο αρχείο script, παρέχεται πρόσβαση σε ένα σύνολο άλλων αρχείων τα οποία απαιτούνται κατά την μεταγλώττιση.

Τέλος, το αρχείο script μπορεί να εκτελείται απευθείας από το λειτουργικό σύστημα, είτε μέσω εξερευνητή αρχείων είτε μέσω συντόμευσης στην επιφάνεια εργασίας, με διπλό αριστερό κλικ.

4.5.4 Οργάνωση αρχείων

Το εκτελέσιμο αρχείο (*.exe*) το οποίο παράγεται μέσω της μεταγλώττισης αποτελεί την εφαρμογή σοβαρού σκοπού και το τελικό αρχείο που θα εκτελούν οι χρήστες με διπλό κλικ. Ωστόσο, δεν είναι το μόνο αρχείο που απαιτείται για να εκτελεστεί η εφαρμογή.

Ιδιαίτερα σε εφαρμογές που προορίζονται για εκτέλεση σε υπολογιστικά συστήματα *άλλων χρηστών*, πρέπει να λαμβάνεται σοβαρά η διαδικασία συγκέντρωσης των αρχείων που απαιτούνται για τη σωστή εκτέλεση. Τα αρχεία αυτά πρέπει να βρίσκονται είτε στον ίδιο κατάλογο με το εκτελέσιμο αρχείο της εφαρμογής (προτιμάται), είτε σε κάποιους βασικούς καταλόγους των Windows όπως το *C:\Windows\system32*, κάτι το οποίο όμως θα απαιτούσε εγκατάσταση της εφαρμογής ώστε να αντιγραφούν τα αρχεία. Κάποια από τα αρχεία που αναφέρονται, υπάρχουν ήδη στο υπολογιστικό σύστημα του χρήστη, ωστόσο αναφέρονται για πληρότητα. Τα αρχεία αυτά, αλλά και η υπόλοιπη οργάνωση αρχείων της εφαρμογής, παρουσιάζονται παρακάτω, με μια σύντομη περιγραφή:

- ✔ **start.exe:** Το εκτελέσιμο αρχείο της εφαρμογής το οποίο παράγεται μέσω της διαδικασίας μεταγλώττισης. Στο αρχείο script της μεταγλώττισης μπορεί να τροποποιηθεί το όνομα του μέσω της μεταβλητής *BINPATH.*

- ✔ **msvcr.dll:** Πρόκειται για αρχείο δυναμικής βιβλιοθήκης, το οποίο απαιτείται από εφαρμογές Windows που έχουν αναπτυχθεί σε C / C++, ανεξάρτητα του μεταγλωττιστή που χρησιμοποιήθηκε για την παραγωγή του εκτελέσιμου αρχείου τους. Ωστόσο, πρόκειται για ένα αρχείο που συνοδεύει όλες τις εκδόσεις των Windows και έτσι δεν απαιτείται η συμπερίληψη του στα αρχεία της εφαρμογής.

- ✔ **msvcr100.dll:** Ότι ισχύει και για το αρχείο *msvcr.dll*. Πρόκειται για μια αναβαθμισμένη έκδοσης της παραπάνω βιβλιοθήκης που συνοδεύει τον μεταγλωττιστή *Visual Studio C++ 2010* της Microsoft. Ωστόσο, η χρήση κάποιων δυναμικών βιβλιοθηκών (όπως το *OpenAL32.dll*), τα αρχεία των οποίων δημιουργήθηκαν μέσω του μεταγλωττιστή *Visual Studio C++*, υποχρεώνουν την εκτελέσιμη εφαρμογή να τα απαιτεί κατά την εκτέλεση.

- ✔ **openGL32.dll:** Πρόκειται για αρχείο δυναμικής βιβλιοθήκης, το οποίο απαιτείται από εφαρμογές Windows, οι οποίες χρησιμοποιούν το API διαχείρισης 3D γραφικών *OpenGL.* Ωστόσο, πρόκειται για ένα αρχείο που συνοδεύει όλες τις εκδόσεις των Windows και έτσι δεν απαιτείται η συμπερίληψη του στα αρχεία της εφαρμογής, ενώ αναβαθμίζεται αυτόματα μέσω της εγκατάστασης οδηγών της κάρτας γραφικών.

- ✔ **openAL32.dll:** Πρόκειται για αρχείο δυναμικής βιβλιοθήκης, το οποίο απαιτείται από εφαρμογές Windows, οι οποίες χρησιμοποιούν το API διαχείρισης 3D ήχου *OpenAL Soft*. Η απαίτηση χρήσης μιας δυναμικής βιβλιοθήκης αντί στατικής (ενσωματωμένη στο εκτελέσιμο αρχείο) έχει να κάνει με τη άδεια χρήσης που έχει οριστεί στο *OpenAL Soft* (LGPL).

- ✔ **data.xml:** Πρόκειται για το αρχείο το οποίο περιέχει το σύνολο του εκπαιδευτικού υλικού της εφαρμογής. Συγκεκριμένα, περιέχονται όλες οι προγραμματιστικές αναθέσεις των χρηστών, χωρισμένων ανά εργασιακή

θέση, καθώς και οι οδηγοί οι οποίοι περιγράφουν τις προγραμματιστικές έννοιες, σε όλες τις υποστηριζόμενες γλώσσες, στις οποίες βασίζονται οι προγραμματιστικές αναθέσεις και τις οποίες οι χρήστες διαβάζουν & εκπαιδεύονται, μέσω της εφαρμογής.

- ✔ **/resource:** Αφορά το folder στο οποίο είναι τοποθετημένοι οι πόροι της εφαρμογής, για παράδειγμα εικόνες, αρχεία ήχου, δείκτες ποντικιού & γραμματοσειρές, κατάλληλα οργανωμένοι, και οι οποίοι πόροι φορτώνονται στο σύνολο τους με την εκτέλεση της εφαρμογής.

- ✔ **/cpp_compiler:** Πρόκειται για το folder του ενσωματωμένου compiler της εφαρμογής για τη γλώσσα προγραμματισμού C++. Μέσω του compiler, η εφαρμογή μπορεί να μεταγλωττίζει, σε πραγματικό χρόνο, τον κώδικα που συντάσσουν οι χρήστες & να παράγεται κατάλληλο feedback στους χρήστες, μέσω του ενσωματωμένου συντάκτη C++ της εφαρμογής.

4.6 Ανάλυση εφαρμογής

Η εκτέλεση της εφαρμογής ξεκινάει από το entry point της γλώσσας προγραμματισμού C++, το οποίο είναι η συνάρτηση `main()`. Παρακάτω, θα ακολουθήσει μια περιγραφική ανάλυση των (σημαντικότερων) ενεργειών που εκτελούνται μέσα στη συνάρτηση `main()` αλλά και πως μοιράζεται ο έλεγχος της εφαρμογής έξω από τη συνάρτηση αυτή.

4.6.1 Εντολές προεπεξεργαστή

Μια παρατήρηση που πρέπει να αναφερθεί στο σημείο αυτό είναι ότι ο κώδικας που θα παρουσιαστεί στη συνέχεια, αφορά *πολλαπλά λειτουργικά συστήματα*. Με άλλα λόγια, συγκεκριμένα κομμάτια κώδικα προορίζονται προς εκτέλεση μόνο σε συγκεκριμένα λειτουργικά συστήματα. Για να επιτευχθεί αυτό, χρησιμοποιούνται εντολές προεπεξεργαστή (αλλιώς και *preprocessor directives*) στον πηγαίο κώδικα της προσαρμοσμένης μηχανής παιχνιδιών.

Οι εντολές αυτές έχουν την μορφή μπλοκ που ανοίγουν και κλείνουν με τις εντολές `#ifdef xxx` και `#endif`, όπου `xxx` μια τιμή, την οποία ελέγχει ο μεταγλωττιστής για την ύπαρξη της ώστε να μεταγλωττίσει μόνο τον κώδικα που περιέχεται στο μπλοκ. Κάνοντας χρήση των μπλοκ αυτών, μπορεί να συμπεριληφθεί κώδικας ο οποίος θα μεταγλωττίζεται – και ουσιαστικά θα είναι ορατός – μόνο από μεταγλωττιστές συγκεκριμένων λειτουργικών συστημάτων και ασφαλής σε άλλους μεταγλωττιστές, καθώς θα αγνοείται τελείως.

Οι τιμές οι οποίες ελέγχονται μέσω της εντολής `#ifdef` αφορούν προγραμματιστικές σταθερές, οι οποίες όμως ορίζονται είτε από τους ίδιους τους μεταγλωττιστές, είτε από βασικά αρχεία του τρέχοντος λειτουργικού συστήματος. Για παράδειγμα, η εντολή `#ifdef _WIN32` ελέγχει εάν η εφαρμογή μεταγλωττίζεται από compiler σε *Windows* περιβάλλον (για παράδειγμα, *Visual Studio* ή *MingGW-w64*), η εντολή `#ifdef __linux__` ελέγχει εάν η εφαρμογή μεταγλωττίζεται από compiler σε *Linux* περιβάλλον (για παράδειγμα, *GCC* ή *MingGW-w64)*.

4.6.2 Κλήσεις API

Στην εργασία παρουσιάζεται αρκετό τμήμα του κώδικα με τον οποίο υλοποιήθηκε η εφαρμογή. Ο κώδικας αυτός περιλαμβάνει τμήματα σε γλώσσα προγραμματισμού C++, κλήσεις συναρτήσεων στο API της εφαρμογής, καθώς και κλήσεις σε άλλα APIs που χρησιμοποιούνται στην εφαρμογή.

Στον Πίνακα 6, περιγράφεται η μορφή των κλήσεων αυτών, ώστε να γίνεται εύκολα αντιληπτό το API στο οποίο ανήκουν και τι σκοπό εξυπηρετούν:

Πίνακας 6: Μορφή κλήσεων API

Κλήσεις API	Περιγραφή
`J3D_xxx()`	Κλήσεις συναρτήσεων στο διαδικαστικό API της *προσαρμοσμένης μηχανής παιχνιδιών Jolt3D*. Παράδειγμα: `J3D_display_set_resolution(...);`
`GAME::xxx()`	Κλήσεις στο αντικειμενοστραφές API του παιχνιδιού. Παράδειγμα: `GAME::load_resources();`

`::xxx()`	Κλήσεις στο διαδικαστικό API *Win32* των Windows. Παράδειγμα: `HWND hwnd = ::CreateWindowEx(...);`
`Xxxx()`	Κλήσεις στο διαδικαστικό API *X11* του Linux. Παράδειγμα: `Display* display = XOpenDisplay(...);`
`Axxx()`	Κλήσεις στο διαδικαστικό API *Android NDK* του Android. Παράδειγμα: `Aasset* asset = AassetManager_open(...);`

`glxxx()`
`eglxxx()`
	Κλήσεις στο διαδικαστικό API γραφικών του *OpenGL*. Παράδειγμα:
	
	`glBindTexture(...);`
	`eglSwapBuffers(...);`

`alxxx()`	Κλήσεις στο διαδικαστικό API ήχου του *OpenAL Soft*. Παράδειγμα: `alListener3f(...);`
`FT_xxx()`	Κλήσεις στο διαδικαστικό API γραμματοσειρών *FreeType*. Παράδειγμα: `FT_Error init = FT_Init_FreeType(...);`

`pugi::xxx`	Κλήσεις στο αντικειμενοστρεφές API διαχείρισης XML αρχείων *PugiXML*. παράδειγμα:
	
	`pugi::xml_document doc;`
	`doc.load_string(...);`

4.6.3 Συνάρτηση J3D_main()

Όπως αναφέρθηκε παραπάνω, η συνάρτηση `main()` είναι η πρώτη συνάρτηση που καλείται (από το λειτουργικό σύστημα) όταν εκτελείται μια εφαρμογή σε C++. Ωστόσο, με χρήση της προσαρμοσμένης μηχανής παιχνιδιών αυτό γίνεται μέσω `J3D_main()`, για τους λόγους που αναφέρθηκαν παραπάνω. Παραδείγματα κλήσεων, σε διάφορα λειτουργικά συστήματα, παρουσιάζονται στον Κώδικα 4:

Κώδικας 4: Κλήση της συνάρτησης J3D_main()

```
#ifdef __ANDROID__
   void android_main(struct android_app* state) {
      ...
      J3D_main();
      exit(0);
   }
#elif __APPLE__ && TARGET_OS_IPHONE)
   int main(int argc, char *argv[]) {
      @autoreleasepool {
         // will eventually call J3D_main() through an event
         return UIApplicationMain(argc, argv, nil,
                                  NSStringFromClass([AppDelegate class])
                                 );
      }
      return 0;
   }
#else
   int main() {
      J3D_main();
      return 0;
   }
#endif
```

Αυτό που συμβαίνει στα παραπάνω παραδείγματα είναι να καλούμε την ειδική συνάρτηση `J3D_main()` της προσαρμοσμένης μηχανής παιχνιδιών όταν ξεκινάει, ουσιαστικά, την εκτέλεση της η εφαρμογή σε κάθε υποστηριζόμενο λειτουργικό σύστημα, δίνοντας έπειτα τον έλεγχο στον προγραμματιστή.

4.6.4 Συνάρτηση J3D_init()

Πρόκειται για την πρώτη κλήση μέσα στην συνάρτηση `J3D_main()` και αφορά την *αρχικοποίηση* της προσαρμοσμένης μηχανής παιχνιδιών. Κάθε λειτουργικό σύστημα παρέχει ένα συγκεκριμένο API, το οποίο η προσαρμοσμένη μηχανή παιχνιδιών χρησιμοποιεί για να διαβάσει ή να εκτελέσει κάποιες βασικές ενέργειες. Μερικές από τις ενέργειες αυτές φαίνονται στον Κώδικα 5, για τα λειτουργικά συστήματα *Windows* & *Linux*:

Κώδικας 5: Τμήμα κώδικα της συνάρτησης J3D_init()

```
Void J3D_init(...) {
   // get desktop resolution
   int desktop_x, desktop_y;
   #ifdef _WIN32
      HWND hDesktop = ::GetDesktopWindow();
      RECT rc;
      ::GetWindowRect(hDesktop, &rc);
      desktop_x = rc.right;
      desktop_y = rc.bottom;
   #elif defined(__linux__) || defined(__sun)
      Display* display = XOpenDisplay(NULL);
      XWindowAttributes attr;
      XgetWindowAttributes(display, XdefaultRootWindow(display), &attr);
      desktop_x = attr.width;
      desktop_y = attr.height;
   #endif
   ...
   FT_Error init = FT_Init_FreeType(...); // freetype initialize
   J3D_AUDIO::init();                     // audio init (OpenAL or OpenSLES)
}
```

Η παραπάνω συνάρτηση έχει κώδικα, προσαρμοσμένο για συγκεκριμένα λειτουργικά συστήματα (μέσω των μπλοκ κώδικα `#ifdef` και `#endif`). Το παραπάνω τμήμα κώδικα – το οποίο αποτελεί μικρό τμήμα της συνάρτησης – δείχνει πως μια συνάρτηση της προσαρμοσμένης μηχανής παιχνιδιών μπορεί να χρησιμοποιήσει κώδικα, ο οποίος παρέχεται από διαφορετικά API λειτουργικών συστημάτων (το Windows *Win32* και το Linux *X11* στην παραπάνω περίπτωση).

Με τον τρόπο αυτό, μπορούν να χρησιμοποιούνται οι ίδιες συναρτήσεις ώστε να πραγματοποιούν τις ίδιες ενέργειες (π.χ. ανάγνωση ανάλυση της οθόνης), ακόμη και σε διαφορετικά λειτουργικά συστήματα, ωστόσο για κάθε υποστηριζόμενο λειτουργικό σύστημα θα εκτελείται διαφορετικός κώδικας καθώς ο μεταγλωττιστής θα έχει φροντίσει να μεταγλωττίσει μόνο το τμήμα κώδικα που αφορά το παρόν λειτουργικό σύστημα. Με τον τρόπο αυτό επιτυγχάνεται η υποστήριξη εφαρμογών σε *πολλαπλά λειτουργικά συστήματα* (*cross-platform*).

Οι κλήσεις συναρτήσεων στο τέλος της συνάρτησης `J3D_init()` αναφέρονται απλά για να δείξουν ότι πραγματοποιείται αρχικοποίηση και των επιπρόσθετων βιβλιοθηκών της προσαρμοσμένης μηχανής παιχνιδιών, όπως είναι η διαχείριση γραμματοσειρών & ήχου.

4.6.5 Συνάρτηση J3D_window_create()

Τα λειτουργικά συστήματα που είναι βασισμένα σε παραθυρικό σύστημα, για παράδειγμα τα *Windows* και το *Mac OSX*, απαιτούν από τις εφαρμογές που αναπτύσσονται σε αυτά και προορίζονται να εμφανίσουν παραθυρικά ή γραφικά στοιχεία, να δημιουργούν τα δικά τους παράθυρα εφαρμογής.

Ωστόσο, το API του κάθε λειτουργικού συστήματος παρέχει συγκεκριμένο τρόπο δημιουργίας παραθύρων αλλά και της διασύνδεσης του με το γραφικό υποσύστημα (στην προκειμένη περίπτωση, το *OpenGL*), κάτι που απαιτείται από την εφαρμογή σοβαρού σκοπού.

Όπως και με τη συνάρτηση `J3D_init()`, χρησιμοποιείται μια ακόμα συνάρτηση της προσαρμοσμένης μηχανής παιχνιδιών, με σκοπό τη δημιουργία του παραθύρου της εφαρμογής σε πολλαπλά λειτουργικά συστήματα, με όνομα `J3D_window_create()`. Οι κύριες εργασίες της συνάρτησης αυτής είναι:

1. δημιουργία του παραθύρου της εφαρμογής
2. ρύθμιση του πλαισίου συσκευής (*device context*) του παραθύρου
3. δημιουργία πλαισίου *OpenGL* & διασύνδεση με το πλαίσιο συσκευής
4. αρχικοποίηση συναρτήσεων (μια φορά) του OpenGL μέσω του GLEW
5. δημιουργία / φόρτωση shaders (μια φορά) στο OpenGL
6. κλήση callback για αρχικοποίηση / φόρτωση αρχείων της εφαρμογής

Οι εργασίες αυτές παρουσιάζονται συνοπτικά στον Κώδικα 6, για τα λειτουργικά συστήματα *Windows* & *Linux*:

Κώδικας 6: Τμήμα κώδικα της συνάρτησης J3D_window_create()

```
void J3D_window_create(...) {
#ifdef _WIN32
   // 1. create window
   WNDCLASSEX wc;
   wc.hInstance = ::GetModuleHandle(NULL);
   wc.lpfnWndProc = J3D_WINDOW::wnd_proc;
   ...
   RegisterClassEx(&wc);
   HANDLE window = ::CreateWindowEx(...);
   // 2. set device context settings
   PIXELFORMATDESCRIPTOR pfd;
```

```
    HDC hdc = ::GetDC( window );
    pfd.nVersion = 1;
    pfd.dwFlags = PFD_DRAW_TO_WINDOW | PFD_SUPPORT_OPENGL | PFD_DOUBLEBUFFER;
    ...
    int32_t iFormat = ::ChoosePixelFormat(hdc, &pfd);
    ::SetPixelFormat(hdc, iFormat, &pfd);
    // 3. create OpenGL context, connect to device context & enable for drawing
    HGLRC context = ::wglCreateContext( hdc );
    ::wglMakeCurrent(hdc, context);
#elif defined(__linux__) || defined(__sun)
    // 1+2. create window & set device context settings
    Glint att[] = {GLX_RGBA, GLX_DEPTH_SIZE, 24, GLX_DOUBLEBUFFER, None};
    XvisualInfo* vi = glXChooseVisual(..., att);
    XsetWindowAttributes attrs;
    attrs.colormap = XcreateColormap(..., vi->visual, ...);
    ...
    Window window = XcreateWindow(.., vi->depth, vi->visual, ..., &attrs);
    ...
    // 3. create OpenGL context, connect to device context & enable for drawing
    GLXContext context = glXCreateContext(..., vi, ...);
    glXMakeCurrent(..., window, context);
#endif
    // 4. initialize GLEW & load all functions found on OpenGL driver
    static bool glew_initialized = false;
    if (!glew_initialized) {
       glewInit();
       glew_initialized = true;
    }
    // 5. create / load shaders on OpenGL
    // 6. callback to initialize / load resources for application
    J3D_callback_window( J3D_WINDOWEVENT(..., J3D_WINDOW_INIT) );
}
```

Το παραπάνω τμήμα κώδικα της συνάρτησης υλοποιεί όλες τις εργασίες που αναφέρθηκαν σε δυο (2) λειτουργικά συστήματα. Οι εργασίες αυτές εμφανίζονται *αριθμημένες* στον κώδικα ώστε να γίνεται αντιστοίχιση με το κείμενο και να υπάρχει μεγαλύτερη κατανόηση της λειτουργίας τους. Υπάρχουν αρκετά τμήματα κώδικα τα οποία λόγω περιπλοκότητας & μεγέθους κώδικα, δεν περιγράφονται καθόλου εδώ π.χ. δημιουργία shaders. Ωστόσο, θα περιγραφούν μεμονωμένα σε επόμενο κεφάλαιο. Η τελευταία γραμμή του κώδικα, μεταφέρει τον έλεγχο της εφαρμογής στην *αρχικοποίηση* της (μέσω callback).

4.6.6 Συνάρτηση J3D_loop()

Στις μοντέρνες εφαρμογές που αναπτύσσονται μέσω κάποιου API γραφικών όπως το OpenGL, απαιτείται κάποιος μηχανισμός ο οποίος θα *καλεί* τον κώδικα της εφαρμογής *ανά τακτά χρονικά διαστήματα* σε έναν *βρόγχο* και ο οποίος θα τερματίζει μόνο με παρέμβαση του χρήστη π.χ. έξοδος.

Οι βασικές λειτουργίες του βρόγχου είναι η διαχείριση της εφαρμογής, τόσο σε επίπεδο λογικής όσο και παρουσίασης γραφικών στην οθόνη. Ωστόσο, το λειτουργικό σύστημα θα πρέπει να *επικοινωνεί* με την εφαρμογή *μέσω μηνυμάτων* κατά τη λειτουργία του βρόγχου, αλλιώς η εφαρμογή δεν θα μπορεί να επεξεργαστεί συγκεκριμένα γεγονότα π.χ. πάτημα ενός πλήκτρου, μετακίνηση του παραθύρου της εφαρμογής κλπ.

Η σχεδίαση & παρουσίαση γραφικών μέσω *OpenGL* πραγματοποιείται μέσω *double buffering* κατά την οποία σχεδιάζονται πρώτα όλα τα γραφικά σε μια *κρυφή* περιοχή, η οποία ονομάζεται *back buffer*, σε κάθε επανάληψη του βρόγχου. Τα γραφικά αυτά δεν φαίνονται στην οθόνη. Στην συνέχεια του βρόγχου και αφού έχουν σχεδιαστεί όλα τα απαιτούμενα γραφικά, *μεταφέρονται* άμεσα στην *ορατή* περιοχή της οθόνης που ονομάζεται *front buffer*.

Μέσω της διαδικασίας αυτής, μπορεί να λειτουργεί *απρόσκοπτα* η σχεδίαση & παρουσίαση γραφικών, χωρίς προβλήματα συγχρονισμού μεταξύ κάρτας γραφικών και μόνιτορ, λόγω *ρυθμού ανανέωσης* της οθόνης. Πιο συγκεκριμένα, οι κύριες εργασίες της συνάρτησης αυτής είναι:

1. επεξεργασία μηνυμάτων του λειτουργικού συστήματος
2. προγραμματισμός *λογικής παιχνιδιού* & σχεδίαση στον *back buffer*
3. μεταφορά των γραφικών στον *front buffer* (εμφάνιση στην οθόνη)

Στον Κώδικα 7 παρουσιάζεται, σε συνοπτική μορφή, ένας τέτοιος βρόγχος, για τα λειτουργικά συστήματα των *Windows* & *Android*:

Κώδικας 7: Τμήμα κώδικα της συνάρτησης J3D_loop()

```
void J3D_loop() {
   while(true) {
#ifdef _WIN32
      MSG msg;
      // 1. check for OS messages
      if (::PeekMessage(&msg, ...)) {
         ::TranslateMessage(&msg);
         ::DispatchMessage(&msg);
         continue;
      }
      // 2. code logic & render graphics on back buffer
      J3D_callback_window( J3D_WINDOWEVENT(..., J3D_WINDOW_LOOP) );
      // 3. copy graphics to front buffer (screen)
      ::SwapBuffers(hdc);
#elif __ANDROID__
      int id, e;   // ident & events
      struct android_poll_source* s;
      // 1. check for OS messages
      while((id=Alooper_pollAll(..., &e,reinterpret_cast<void**>(&s))) >=0) {
         if (s != NULL) {
            s->process(..., s);
         }
      }
      // 2. code logic & render graphics on back buffer
      J3D_callback_window( J3D_WINDOWEVENT(..., J3D_WINDOW_LOOP) );
      // 3. copy graphics to front buffer (screen)
      eglSwapBuffers(...);
#endif
   }
}
```

Το παραπάνω τμήμα κώδικα της συνάρτησης υλοποιεί όλες τις εργασίες που αναφέρθηκαν σε δυο (2) λειτουργικά συστήματα. Οι εργασίες αυτές επαναλαμβάνονται για κάθε στιγμιότυπο (*frame)* του βρόγχου της εφαρμογής. Σε κάθε στιγμιότυπο, γίνεται κλήση στην συνάρτηση `J3D_callback_window()` της προσαρμοσμένης μηχανής παιχνιδιών, όπου και βρίσκεται συγκεντρωμένη η *προγραμματισμένη λογική* του παιχνιδιού.

4.6.7 Shaders

Στις μοντέρνες εφαρμογές που αναπτύσσονται σε OpenGL, είναι συχνή η χρήση μικρών τμημάτων κώδικα, τα οποία όμως εκτελούνται αποκλειστικά στον επεξεργαστή της *κάρτας γραφικών* (GPU). Τα τμήματα αυτά κώδικα γράφονται σε ιδιαίτερες γλώσσες προγραμματισμού που λέγονται *γλώσσες σκίασης* ή shading languages. Κύρια εργασία τους είναι η επεξεργασία δεδομένων 3D γραφικών, με προγραμματιστικό τρόπο.

Στο *OpenGL*, η κύρια γλώσσα σκίασης είναι η *GLSL*. Είναι μια υψηλού επιπέδου γλώσσα προγραμματισμού, cross-platform, ενώ το συντακτικό της θυμίζει την γλώσσα C. Η αρχική έκδοση της *GLSL* παρουσιάστηκε το 2004 μέσω της έκδοσης 2.0 του *OpenGL* οπότε και ξεκίνησε, άτυπα, η ανάπτυξη μοντέρνων εφαρμογών *OpenGL*, με προγραμματισμό και της GPU.

Πριν την έλευση των γλωσσών σκίασης και των καρτών γραφικών που τις υποστηρίζουν, η ανάπτυξη των *OpenGL* εφαρμογών δεν βασιζόταν σε shaders, αλλά στο ίδιο το API του *OpenGL*. Υπήρχαν – και υπάρχουν ακόμη – εντολές μέσα στο *OpenGL* API, με σημαντική χρήση της CPU, με τις οποίες ένας προγραμματιστής μπορεί να σχεδιάσει 3D γραφικά π.χ. `glVertex3f(...)` ή να ορίσει ρυθμίσεις φωτισμού π.χ. `glLightf(...)`.

Οι εντολές αυτές πρωτοεμφανίστηκαν σε μια εποχή όπου ο κεντρικός επεξεργαστής (CPU) καθόριζε την απόδοση, κάτι που ήταν αναμενόμενο καθώς η συντριπτική πλειοψηφία των εφαρμογών ήταν 2D. Ωστόσο, αν και είναι άμεσα κατανοητές και εύκολες στη χρήση, δεν βοηθούν στην σημερινή εποχή των τεχνολογικά εξελιγμένων καρτών γραφικών και των απαιτητικών παιχνιδιών που τις αξιοποιούν, όπου υπάρχει αυξημένη ανάγκη για μεγαλύτερη χρήση της GPU έναντι της CPU. Αυξημένη χρήση της GPU σημαίνει και προγραμματισμός της.

Στην προσαρμοσμένη μηχανή παιχνιδιών που χρησιμοποιεί η εφαρμογή, γίνεται χρήση δυο (2) shaders. Ο ένας shader χρησιμοποιείται για να ρυθμίζει αυτόματα την εφαρμογή σε περιβάλλον *πλήρους οθόνης* (fullscreen), ενώ ο άλλος shader χρησιμοποιείται για την σχεδίαση 2D γραφικών της εφαρμογής. Οι shaders αυτοί υπάρχουν σε μορφή κώδικα μέσα στην εφαρμογή (απλό κείμενο) και μεταγλωττίζονται – σε πραγματικό χρόνο και στον κατάλληλο κώδικα μηχανής που απαιτεί η GPU – όταν εκτελείται η εφαρμογή.

Την φόρτωση, μεταγλώττιση και χρήση των shaders την πραγματοποιεί το API του OpenGL. Στον Κώδικα 8 παρουσιάζεται, σε συνοπτική μορφή, ο vertex shader που χρησιμοποιείται για την διαχείριση των vertices των γεωμετρικών σχημάτων, πριν αυτά μετασχηματιστούν, μέσω του fragment shader, σε pixels στην οθόνη:

Κώδικας 8: Τμήμα κώδικα shader & χρήσης του από OpenGL

```
string shadertext =
    "attribute vec3 aVertexPosition;    \n"
    "attribute vec4 aVertexColor;       \n"
    "attribute vec2 aTextureCoord;      \n"
    "uniform mat4 uPxMVMatrix;          \n"
    "varying vec4 vVertexColor;         \n"
    "varying vec2 vTextureCoord;        \n"
    "void main() {                      \n"
    "   vVertexColor = aVertexColor;    \n"
    "   vTextureCoord = aTextureCoord;  \n"
    "   gl_Position = uPxMVMatrix * vec4(aVertexPosition, 1.0);          \n"
    "}                                                                   \n";


    int32_t program = glCreateProgram();
    GLuint shader = glCreateShader(GL_VERTEX_SHADER);
    glShaderSource(shader, 1, &shadertext.c_str(), NULL);
    glCompileShader(shader);
    ...
    glUseProgram( program );
```

Το πρώτο βήμα που γίνεται είναι να συγκεντρωθεί ο κώδικας του vertex shader σε μια μεταβλητή string. Αυτό μπορεί να γίνει με διάφορους τρόπους π.χ. φόρτωση από εξωτερικό αρχείο. Στη συνέχεια, δημιουργείται ο shader που θα χρησιμοποιείται στην εφαρμογή, μέσω της συνάρτησης `glCreateProgram()`. Ωστόσο, ο shader πρέπει πρώτα να μεταγλωττιστεί σε κατάλληλο κώδικα μηχανής – και σε πραγματικό χρόνο – από το OpenGL και να φορτωθεί στην κάρτα γραφικών, το οποίο γίνεται στην συνάρτηση `glCompileShader(...)`.

Τέλος, ο shader μπορεί να χρησιμοποιείται από την εφαρμογή – όποτε είναι αυτό επιθυμητό – μέσω της εντολής `glUseProgam(...)`. Όλες οι εντολές που θα ακολουθήσουν, θα χρησιμοποιούν το τρέχων πρόγραμμα shader.

4.7 Διαχείριση περιεχομένου

Η εφαρμογή σοβαρού σκοπού χρησιμοποιεί δυο (2) κύριους τρόπους για τη διαχείριση του περιεχομένου, τόσο από πλευράς σύνταξης κώδικα του χρήστη όσο και παρουσίασης εκπαιδευτικού περιεχομένου στον χρήστη.

Ο πρώτος τρόπος διαχείρισης περιεχομένου αφορά στη χρήση του ενσωματωμένου μεταγλωττιστή C++, μέσω του οποίου ο κώδικας που έχει συνταχθεί από τον χρήστη στον συντάκτη C++ της εφαρμογής, μεταγλωττίζεται, εκτελείται & τελικά επαληθεύεται για ορθότητα του. Ο δεύτερος τρόπος διαχείρισης περιεχομένου αφορά στη παρουσίαση του εκπαιδευτικού περιεχομένου στον χρήστη, μέσω διαχείρισης XML αρχείων.

4.7.1 Μεταγλώττιση κώδικα στον συντάκτη C++

Η εφαρμογή σοβαρού σκοπού παρέχει έναν ιδιαίτερα προσαρμοσμένο κειμενογράφο στον χρήστη ώστε να μπορεί να συντάσσει τον διορθωμένο κώδικα του για τις προγραμματιστικές εργασίες που του ανατίθενται. Ωστόσο, ο κώδικας αυτός θα πρέπει να μεταγλωττίζεται, εκτελείται & τελικά να επαληθεύεται για την ορθότητα του, προτού αξιολογηθεί ο διορθωμένος κώδικας.

Στον Κώδικα 9, παρουσιάζεται ο ολοκληρωμένος κώδικας που χρησιμοποιεί η εφαρμογή για την μεταγλώττιση, την διασύνδεση (linking), την εκτέλεση & τελικά την παραγωγή εξόδου, στο τμήμα κώδικα που σύνταξε ο χρήστης μέσω του συντάκτη C++.

Κώδικας 9: Κώδικας της διαδικασίας μεταγλώττισης του συντάκτη C++

```
// 1. write the user's code to disk as 'code.cpp'
J3D_file_write("code.cpp", code, L"", false);
// 2. compile 'code.cpp' using Digital Mars C/C++ compiler
string compile_output = J3D_shell_exec(
  "cpp_compiler\\bin\\dmc.exe code.cpp -c -v0 -Icpp_compiler\\stlport\stlport"
);
// 3. link compiled 'code.obj' to the executable file 'code.exe'
compile_output += J3D_shell_exec("cpp_compiler\\bin\\link.exe code.obj");
// 4. remove the non-necessary files produced by the compiler
if (J3D_file_exists("code.cpp")) { J3D_file_delete("code.cpp"); }
if (J3D_file_exists("code.map")) { J3D_file_delete("code.map"); }
if (J3D_file_exists("code.obj")) { J3D_file_delete("code.obj"); }
string exec_output;
```

```
if (J3D_file_exists("code.exe")) {
   // 5. execute the executable file 'code.exe' & get its output
   exec_output = J3D_shell_exec("code.exe");
}
```

Το τελικό ζητούμενο από τον παραπάνω κώδικα είναι η μεταγλώττιση του κώδικα του χρήστη και η αποθήκευση της παραγόμενης εξόδου, τόσο από την μεταγλώττιση όσο και από την εκτέλεση του αρχείου του χρήστη. Τα νούμερα που εμφανίζονται στον κώδικα, αντιστοιχίζονται στον πίνακα παρακάτω. Συνοπτικά, οι ενέργειες οι οποίες επιτελούνται στον κώδικα της μεταγλώττισης, είναι οι παρακάτω:

1. **Μεταφορά του κώδικα σε μορφή αρχείου:** Ο κώδικας του χρήστη μεταφέρεται σε μορφή αρχείου με όνομα *code.cpp* ώστε να μπορεί να το επεξεργαστεί κατάλληλα ο ενσωματωμένος compiler.

2. **Μεταγλώττιση του αρχείου σε γλώσσα μηχανής:** Ο κώδικας μεταγλωττίζεται σε ενδιάμεσο αρχείο *.obj* γλώσσας μηχανής, ωστόσο δεν μπορεί να εκτελεστεί καθώς πρέπει πρώτα να *συνδεθεί* με απαραίτητες βιβλιοθήκες τις οποίες χρησιμοποιεί η εφαρμογή, το οποίο κάνει ο *linker*.

3. **Διασύνδεση του αρχείου κώδικα μηχανής με βιβλιοθήκες:** Το βήμα αυτό παράγει το τελικό εκτελέσιμο πρόγραμμα *code.exe*, κατάλληλο για εκτέλεση σε περιβάλλον Windows.

4. **Διαγραφή των περιττών αρχείων της μεταγλώττισης:** Σε μια διαδικασία μεταγλώττισης, παράγεται ένας αριθμός από διάφορα αρχεία τα οποία απαιτούνται κατά την διαδικασία, ωστόσο μετά την δημιουργία του εκτελέσιμου αρχείου δεν είναι (συνήθως) χρήσιμα. Τα αρχεία αυτά διαγράφονται σε αυτό το βήμα, καθώς η εφαρμογή χρειάζεται μόνο το *τελικό εκτελέσιμο αρχείο* του κώδικα του χρήστη.

5. **Εκτέλεση του αρχείου code.exe:** Ο κώδικας που σύνταξε ο χρήστης εκτελείται σε αυτό το βήμα, μέσω ειδικής συνάρτησης της προσαρμοσμένης μηχανής παιχνιδιών με όνομα `J3D_shell_exec()`. Η

συνάρτηση αυτή εκτελεί το εκτελέσιμο που υπάρχει στην παράμετρο του, μαζί με τις παραμέτρους που αυτό φέρει. Ωστόσο, αυτό το κάνει στο *προσκήνιο* και έτσι ο χρήστης δεν αντιλαμβάνεται την διαδικασία. Τέλος, η έξοδος που παράγει το τμήμα κώδικα του χρήστη είναι απαραίτητη, καθώς είναι αυτό που χρησιμοποιείται για την αξιολόγηση του κώδικα σε σχέση με το ζητούμενο της προγραμματιστικής ανάθεσης.

4.7.2 Εκπαιδευτικό περιεχόμενο

Το σύνολο του εκπαιδευτικού περιεχομένου της εφαρμογής περιέχεται σε ένα αρχείο με όνομα *data.xml.* Το πρότυπο αυτό (*XML*) επιλέχθηκε για τους παρακάτω λόγους:

- ✔ **Οργάνωση:** Τα αρχεία XML είναι βασισμένα σε μη-προκαθορισμένα tags (ετικέτες) που ορίζει ο δημιουργός, κάθε μια από τις οποίες μπορεί να περιέχει τα δεδομένα σε οργανωμένη μορφή.

- ✔ **Υποστήριξη Unicode:** Τα αρχεία XML, καθώς και η βιβλιοθήκη που χρησιμοποιείται από την εφαρμογή για την επεξεργασία τους (*PugiXML*) υποστηρίζουν την κωδικοποίηση χαρακτήρων *Unicode*, συγκεκριμένα το πρότυπο *UTF-8*. Η κωδικοποίηση αυτή επιτρέπει την εισαγωγή κειμένου στο XML, το οποίο είναι γραμμένο σε *οποιαδήποτε γλώσσα*, χωρίς να απαιτεί την υποστήριξη της γλώσσας από το λειτουργικό σύστημα.

- ✔ **Ευκολία διαχείρισης:** Τα αρχεία XML επεξεργάζονται εύκολα, με την χρήση ενός απλού κειμενογράφου. Δεν απαιτείται επιπλέον εφαρμογή για την διαχείριση τα αρχείων αυτών, κάνοντας τα προσβάσιμα σε όλους.

- ✔ **Επεκτασιμότητα:** Η εφαρμογή διαβάζει απλώς τα περιεχόμενα που περιέχονται στο αρχείο XML και τα επεξεργάζεται στην εφαρμογή. Θα μπορούσαν πολύ εύκολα να προστεθούν νέες προγραμματιστικές αναθέσεις, για συγκεκριμένη εργασιακή θέση αλλά και για συγκεκριμένη γλώσσα, με απλή επεξεργασία του XML αρχείου.

- ✔ **Αποδοχή:** Τα πρότυπο XML είναι καθολικά αποδεκτό. Χρησιμοποιείται από εφαρμογές σε εκτελέσιμη μορφή, διαδικτυακές εφαρμογές, ως δυναμικές βάσεις δεδομένων ή και ρυθμίσεις εφαρμογών.

Στον Κώδικα 10, παρουσιάζεται ένα μικρό τμήμα του αρχείου *data.xml* και την μορφή που περιέχει την πληροφορία. Συγκεκριμένα, παρουσιάζεται ο κώδικας που περιγράφει την *πρώτη* προγραμματιστική ανάθεση για την θέση *Entry Level*. Στον Κώδικα 11, παρουσιάζεται μια συνοπτική μορφή του κώδικα που χρησιμοποιείται στην εφαρμογή για την ανάγνωση των περιεχομένων του αρχείου *data.xml*, προτού αυτά επεξεργαστούν & εμφανιστούν στην εφαρμογή:

Κώδικας 10: Τμήμα κώδικα στο data.xml

```
<?xml version="1.0"?>
<task id="1" position="0" guide="main,comments">
 <code>
  <![CDATA[
int main() {
   return 0;
}
// We need some does-nothing
// code for a beginner tutorial!
// Please, type and print.
  ]]>
 </code>
 <correct_output>
  <![CDATA[]]>
 </correct_output>
</task>
...
```

Κώδικας 11: Τμήμα κώδικα ανάγνωσης XML

```
// read a task from XML data + add it as a task
pugi::xml_document doc;
string xmlpath = "data.xml";
string xmldata = J3D_file_read( xmlpath );
pugi::xml_parse_result result = doc.load_string( xmldata.c_str() );
if (result.status != pugi::status_ok) {
   ...
   return;
}
// read specific task from XML data
int32_t task_id = task.get_next_task_id_in_xml();
pugi::xml_node thetask = doc.find_child_by_attribute(
   "task", "id", J3D_string_format("%d",task_id).c_str()
);
// no more tasks of this position; out
int32_t position = thetask.attribute("position").as_int();
if (position != GAME::Employee::get_position()) {
   return;
}
string code = string( thetask.child("code").text().as_string() );
...
```

5 Παρουσίαση του Παιχνιδιού

5.1 Εισαγωγή

Η εφαρμογή υλοποιήθηκε με μια λογική *σελίδων* & των *καταστάσεων* τους. Κάθε σελίδα της εφαρμογής μπορεί να αποτελεί μια κλάση από μόνη της (για παράδειγμα, η εισαγωγή της εφαρμογής), ενώ υπάρχουν σελίδες που κάνουν χρήση πολλών κλάσεων (π.χ. το εικονικό γραφείο του προγραμματιστή).

Στο κεφάλαιο αυτό θα παρουσιαστεί & θα αναλυθεί κάθε *σελίδα* της εφαρμογής & οι *καταστάσεις* στις οποίες χωρίζεται. Για κάθε σελίδα της εφαρμογής, θα παρουσιάζονται σχετικές εικόνες & σημαντικά τμήματα κώδικα ώστε να υπάρχει μεγαλύτερη κατανόηση της υλοποίησης.

Η όλη λογική της ιδέας & χρήσης των *σελίδων* αποσκοπεί ώστε κάθε διαφορετική λειτουργικότητα που εμπεριέχεται σε μια *σελίδα* να οδηγεί στο τέλος σε άλλη σελίδα, αλλά και να προέρχεται από κάποια άλλη σελίδα, δίνοντας της εντύπωση ξεφυλλίσματος ενός *βιβλίου.* Κάθε σελίδα χωρίζεται σε *καταστάσεις*, οι οποίες προσαρμόζουν, τόσο οπτικά, όσο και λειτουργικά, την λειτουργία της κάθε σελίδας. Αυτό οδηγεί σε απλούστευση της *λογικής σχεδίασης* παιχνιδιού.

Συνοπτικά, οι *σελίδες* από τις οποίες αποτελείται η εφαρμογή, καθώς και μια μικρή περιγραφή τους, είναι οι παρακάτω (με σειρά εμφάνισης στην εφαρμογή):

1. **Τίτλος:** Η σελίδα αυτή υλοποιεί την γραφική εισαγωγή της εφαρμογής. Οδηγεί στην σελίδα *Φόρμα αίτησης για εργασία*.

2. **Φόρμα αίτησης για εργασία:** Η σελίδα αυτή υλοποιεί, πρακτικά, την εισαγωγή δεδομένων από την πλευρά του χρήστη π.χ. όνομα & θέση. Οδηγεί στην σελίδα *Κανόνες / Οδηγίες*.

3. **Κανόνες / Οδηγίες**: Περιγράφει οδηγίες προς τον χρήστη, όπως τον έλεγχο εισόδου π.χ. κουμπιά, ποντίκι και τις διάφορες λειτουργίες που επιτελούν, αλλά και τους κανόνες στο εργασιακό περιβάλλον του χρήστη. Οδηγεί στην σελίδα *Γραφείο*.

4. **Γραφείο:** Αποτελεί την κεντρική οθόνη του παιχνιδιού και στην οποία *συμβαίνουν* όλα. Εδώ ο χρήστης δέχεται τις εργασιακές του αναθέσεις, συντάσσει κώδικα & εκπαιδεύεται στον προγραμματισμό

5.2 Τίτλος

Με την εκτέλεση της εφαρμογής, ο έλεγχος μεταφέρεται στην σελίδα του τίτλου της εφαρμογής. Εδώ υπάρχει μια σειρά από γραφικά εφέ, ξεκινώντας από το *σταδιακό* γράψιμο του ονόματος της εφαρμογής σε στυλ πληκτρολόγησης και καταλήγοντας στη σταδιακή εμφάνιση των γραφικών του τίτλου.

Ο χρήστης μπορεί να παρακάμψει το σύνολο της εισαγωγής, πατώντας το *Enter* όταν αυτό εμφανιστεί στην οθόνη, μετά από μερικά δευτερόλεπτα, κάτι που είναι βολικό μετά από αρκετές χρήσεις της εφαρμογής. Ο χρήστης επιλέγει την γλώσσα την οποία επιθυμεί από ένα σύνολο δυο (2) γλωσσών, Ελληνικά και Αγγλικά. Το σύνολο της εφαρμογής θα εμφανίζεται στην επιλεγμένη γλώσσα. Στην Εικόνα 9, παρουσιάζεται η ολοκληρωμένη μορφή της οθόνης τίτλου:

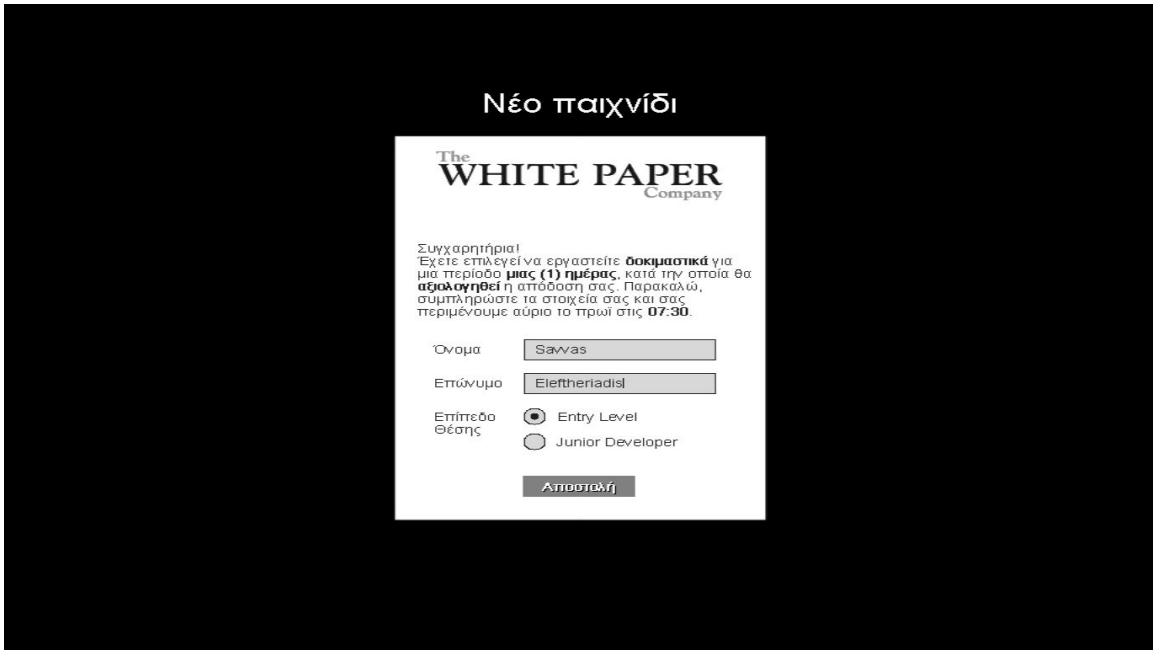
Εικόνα 9: Office Madness - Οθόνη τίτλου

5.3 Φόρμα αίτησης για εργασία

Η σελίδα αυτή εμφανίζεται μετά την εισαγωγή. Αποτελεί ένα είδος *εισαγωγής στοιχείων*, τα οποία θα χρησιμοποιηθούν από την εφαρμογή, όπως για παράδειγμα το όνομα & επώνυμο του χρήστη, καθώς και τη θέση εργασίας στην οποία επιθυμεί να εργαστεί.

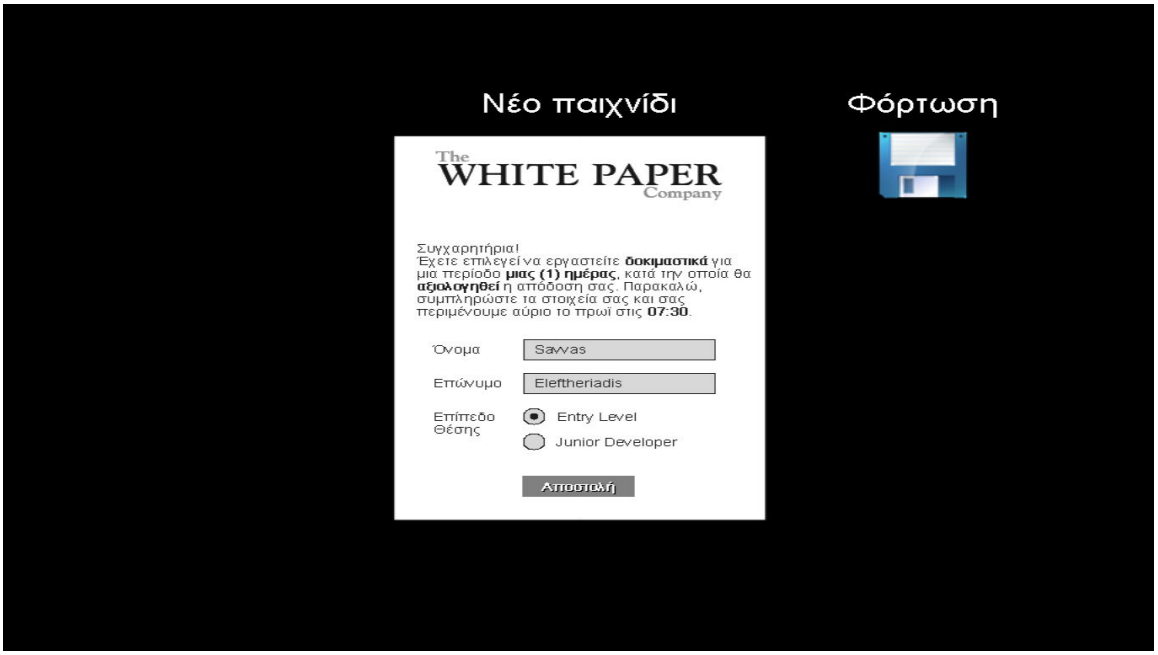
Το ονοματεπώνυμο δεν είναι ουσιαστική πληροφορία & χρησιμοποιείται απλά ως γραφικό στοιχείο επάνω στο γραφείο, κάτι που δεν συμβαίνει όμως με την επιλογή θέσης εργασίας. Εάν ο χρήστης επιλέξει ως επιθυμητή θέση το *Entry Level*, τότε οι εργασιακές αναθέσεις θα είναι από βασικές μέχρι σχετικά εύκολες. Ωστόσο, στο επίπεδο *Junior developer*, η δυσκολία των εργασιακών αναθέσεων αυξάνει σημαντικά. Η επιλογή αυτή παίζει τον ρόλο της επιλογής *δυσκολίας* του παιχνιδιού, με βάση τις υφιστάμενες γνώσεις των χρηστών.

Το σενάριο θέλει τον υποθετικό εργαζόμενο να έχει στείλει, κάποια στιγμή, μια πραγματική αίτηση για εργασία στην υποτιθέμενη εταιρεία και η φόρμα εισαγωγής στοιχείων είναι πρακτικά η απάντηση της εταιρείας, η οποία ζητάει από τον εργαζόμενο να συμπληρώσει κάποια στοιχεία, ενημερώνοντας τον ότι την επόμενη ημέρα θα τον δοκιμάσει σε συνθήκες πραγματικής εργασίας. Στην Εικόνα 10, παρουσιάζεται η οθόνη της φόρμας αίτησης για εργασία:



Εικόνα 10: Office Madness - Οθόνη φόρμας αίτησης για εργασία

Η νέα αίτηση ξεκινάει το παιχνίδι από την αρχή. Ωστόσο, ο χρήστης έχει τη δυνατότητα να αποθηκεύσει την πρόοδο του κατά την χρήση της εφαρμογής. Εάν ο χρήστης έχει αποθηκευμένο παιχνίδι, θα εμφανιστεί επιλογή φόρτωσης του παιχνιδιού του στην ίδια σελίδα. Στην Εικόνα 11, παρουσιάζεται η επιλογή φόρτωσης αποθηκευμένου παιχνιδιού:



Εικόνα 11: Office Madness – Οθόνη φόρτωσης αποθηκευμένου παιχνιδιού

5.4 Κανόνες / Οδηγίες

Η οθόνη αυτή πληροφορεί τον χρήστη για τους κανονισμούς της εταιρείας & με τι θα ασχολείται, καθώς και τον γενικότερο χειρισμό της εφαρμογής. Αφορά μια οθόνη καθαρά ενημερωτική, προτού μπει στο εικονικό του γραφείο και αρχίσει την ολοκλήρωση των εργασιών του.

Στην Εικόνα 12 παρουσιάζεται η οθόνη κανονισμών της εταιρείας, ενώ στην Εικόνα 13, παρουσιάζεται ο χειρισμός της εφαρμογής, σε επίπεδο πλήκτρων & ενέργειας η οποία επιτελείται:

Εικόνα 12: Office Madness - Οθόνη κανονισμών της εταιρείας

Εικόνα 13: Office Madness – Οθόνη χειρισμού

5.5 Γραφείο

Η κύρια οθόνη του παιχνιδιού στην οποία εκτυλίσσεται η εφαρμογή. Εμφανισιακά, αποτελεί το εικονικό γραφείο του εργαζόμενου προγραμματιστή, τον οποίο χειρίζεται ο χρήστης. Το κύριο εργαλείο εργασίας του αποτελεί η οθόνη, στην οποία περιέχονται και όλες οι εφαρμογές που θα χρησιμοποιήσει για να ολοκληρώσει τις εργασιακές του αναθέσεις. Η οθόνη αυτή περιέχει αρκετές λειτουργίες, οι οποίες αναλύονται περιγραφικά παρακάτω.

5.5.1 Tutorial

Την πρώτη λειτουργία του γραφείου αποτελεί το ολοκληρωμένο tutorial, μεταφρασμένο σε όλες τις υποστηριζόμενες γλώσσες, το οποίο επεξηγεί περιγραφικά όλες τις λειτουργίες του γραφείου, βήμα προς βήμα, εστιάζοντας με γραφικό τρόπο στο σημείο ενδιαφέροντος.

Το tutorial μπορεί να παρακαμφθεί μετά από κάποια δευτερόλεπτα, το οποίο είναι χρήσιμο εάν ο χρήστης έχει μάθει την εφαρμογή ύστερα από κάποιες χρήσεις. Στην Εικόνα 14 παρουσιάζεται ένα στιγμιότυπο του tutorial, εξηγώντας μια κύρια λειτουργία.

Εικόνα 14: Office Madness – Στιγμιότυπο της οθόνης tutorial

5.5.2 Οθόνη

Το μόνιτορ του γραφείου αποτελεί το κύριο εργαλείο του εργαζόμενου. Μέσω της οθόνης μόνιτορ ολοκληρώνει τις εργασιακές του αναθέσεις αλλά και εκπαιδεύεται σε νέες προγραμματιστικές έννοιες ή κάνει επανάληψη παλιότερες.

Η οθόνη ανοίγει μέσω του κουμπιού κάτω-δεξιά, κάτι που επεξηγείται και στο tutorial. Τρεις (3) είναι οι εφαρμογές με τις οποίες ασχολείται ο εικονικός εργαζόμενος, ο Σ*υντάκτης C++, οι Οδηγοί προγραμματιστικών εννοιών* και η *Αποθήκευση προόδου*.

Ο *Συντάκτης C++*, ο οποίος αποτελεί μια απλουστευμένη μορφή ενός ολοκληρωμένου IDE, με δυνατότητα μεταγλώττισης, εκτελείται κάνοντας κλικ στο αντίστοιχο εικονίδιο. Οι *Οδηγοί προγραμματιστικών εννοιών* ανοίγουν επίσης με κλικ, όπου παρουσιάζονται όλοι οι διαθέσιμοι οδηγοί σε μορφή εικονικού PDF.

Τέλος, η *Αποθήκευση προόδου* αποθηκεύει την τρέχουσα πρόοδο του χρήστη (ολοκληρωμένες αναθέσεις, αξιολογήσεις, ώρα κλπ.). Ο χρήστης μπορεί να αποθηκεύσει, κλείνοντας την τρέχουσα εφαρμογή (εάν υπάρχει) και κάνοντας κλικ στο εικονίδιο της επιφάνειας εργασίας επάνω δεξιά (δισκέτα). Εάν υπάρχει αποθηκευμένη πρόοδος στο παιχνίδι, θα υπάρχει επίσης και αντίστοιχη επιλογή φόρτωσης στην σελίδα *Φόρμα αίτηση για εργασία*.

Μόνο μια εφαρμογή μπορεί να εκτελείται ανά πάσα στιγμή. Έτσι, εάν εκτελείται ήδη μια από τις εφαρμογές, θα πρέπει να την κλείσουμε πρώτα και ύστερα να ανοίξουμε την επιθυμητή. Στην Εικόνα 15 παρουσιάζεται η επιφάνεια εργασίας μιας ανοιχτής οθόνης μόνιτορ, χωρίς ωστόσο να εκτελείται ακόμη κάποια εφαρμογή.

Εικόνα 15: Office Madness – Επιφάνεια εργασίας οθόνης

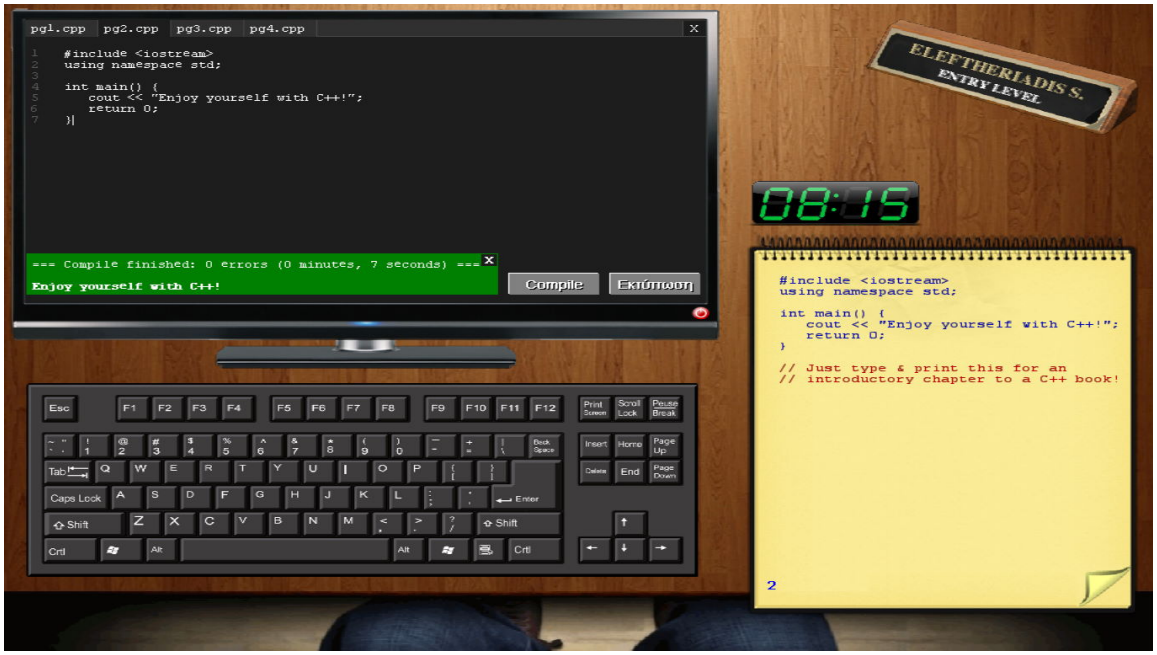
5.5.3 Συντάκτης C++

Αποτελεί, με την πρώτη ματιά, έναν απλό κειμενογράφο στον οποίο ο χρήστης μπορεί να γράφει τον διορθωμένο κώδικα C++ των προγραμματιστικών εργασιών που του ανατίθενται από την εταιρεία, οι οποίες εργασίες συνήθως έχουν συντακτικά προβλήματα.

Ωστόσο, η σύνταξη του διορθωμένου κώδικα είναι η μια λειτουργία. Η δεύτερη λειτουργία του συντάκτη αφορά την *μεταγλώττιση του κώδικα* που έχει γράψει ο χρήστης, σε *εκτελέσιμο πρόγραμμα* και σε *πραγματικό χρόνο*, με το πάτημα του αντίστοιχου κουμπιού που βρίσκεται κάτω δεξιά του συντάκτη. Επομένως, ο συντάκτης C++ μετατρέπεται ουσιαστικά σε ένα IDE, ικανό να μεταγλωττίζει και να επαληθεύει κώδικα σε C++ για την ορθότητα του.

Η διαδικασία της μεταγλώττισης χρησιμοποιεί τον ενσωματωμένο μεταγλωττιστή C++ της εταιρείας Digital Mars. Περισσότερες πληροφορίες υπάρχουν στο κεφάλαιο *4.2.4 Digital Mars C/C++ Compiler*. Συνοπτικά, η διαδικασία της μεταγλώττισης εκτελεί τον compiler με το παρόν κώδικα που έχει συντάξει ο χρήστης και τον μετατρέπει σε ενδιάμεσο κώδικα μηχανής (.obj αρχείο). Στην συνέχεια, εκτελείται ο linker ο οποίος διασυνδέει το παραπάνω αρχείο *.obj* με όλες τις απαιτούμενες βιβλιοθήκες και παράγεται το τελικό

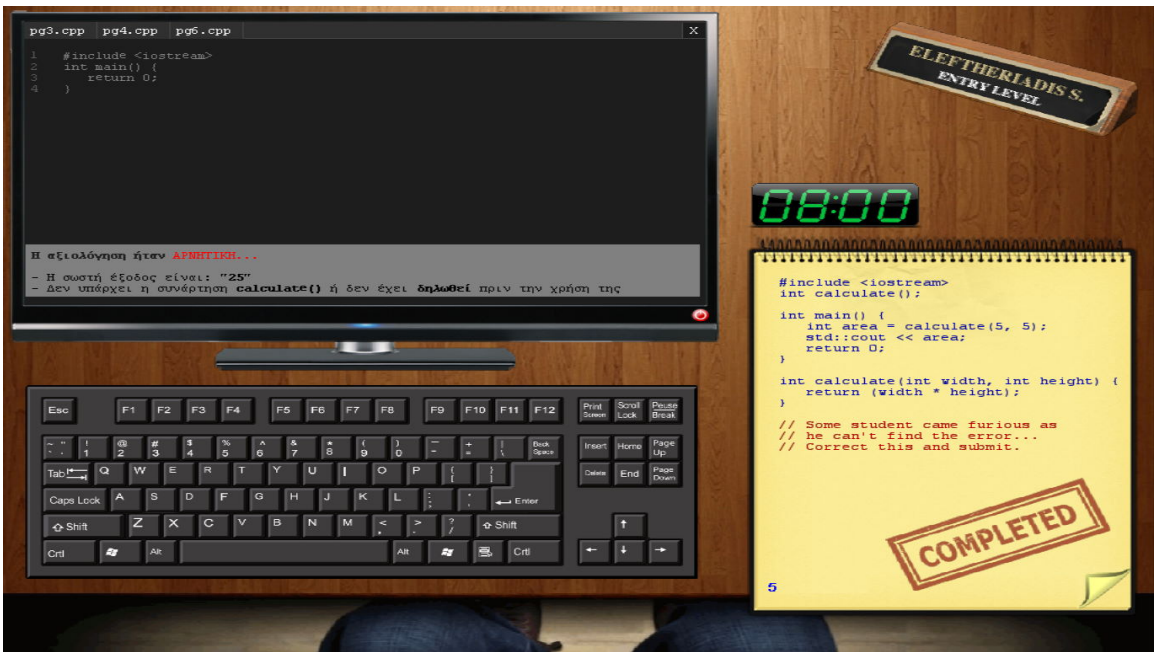
εκτελέσιμο αρχείο του κώδικα του χρήστη. Τέλος, το αρχείο αυτό εκτελείται και επιστρέφεται, δυναμικά, η έξοδος που παράγει ώστε να παρουσιαστεί στον χρήστη με τη μορφή κειμένου. Στην Εικόνα 16 παρουσιάζεται μια τυπική μεταγλώττιση σε ένα τμήμα κώδικα που σύνταξε ο χρήστης, σε μια ανάθεση προγραμματιστικής εργασίας:



Εικόνα 16: Office Madness – Μεταγλώττιση στον συντάκτη C++

Μετά την διαδικασία της μεταγλώττισης & εξόδου του κώδικα που έγραψε ο χρήστης, αξιολογείται ο κώδικας του για ορθότητα, με βάση το ζητούμενο της προγραμματιστικής ανάθεσης. Το αποτέλεσμα είναι είτε θετικό είτε αρνητικό. Εάν είναι αρνητικό, τότε εμφανίζονται ένας ή παραπάνω λόγοι που αυτό συνέβη.

Η συγκεκριμένη ανάδραση που παρέχεται στον χρήστη, θα τον βοηθήσει να κατανοήσει σε επίπεδο κώδικα τα λάθη του. Οι προγραμματιστικές αναθέσεις οι οποίες ολοκληρώνονται, δεν αφαιρούνται από την λίστα των αναθέσεων. Αντιθέτως, εμφανίζονται με αλλαγμένη μορφή ώστε ο χρήστης να μπορεί να διαπιστώσει, ανά πάσα στιγμή, γιατί αξιολογήθηκε αρνητικά η λύση που έδωσε. Στην Εικόνα 17, παρουσιάζεται μια αρνητική αξιολόγηση:



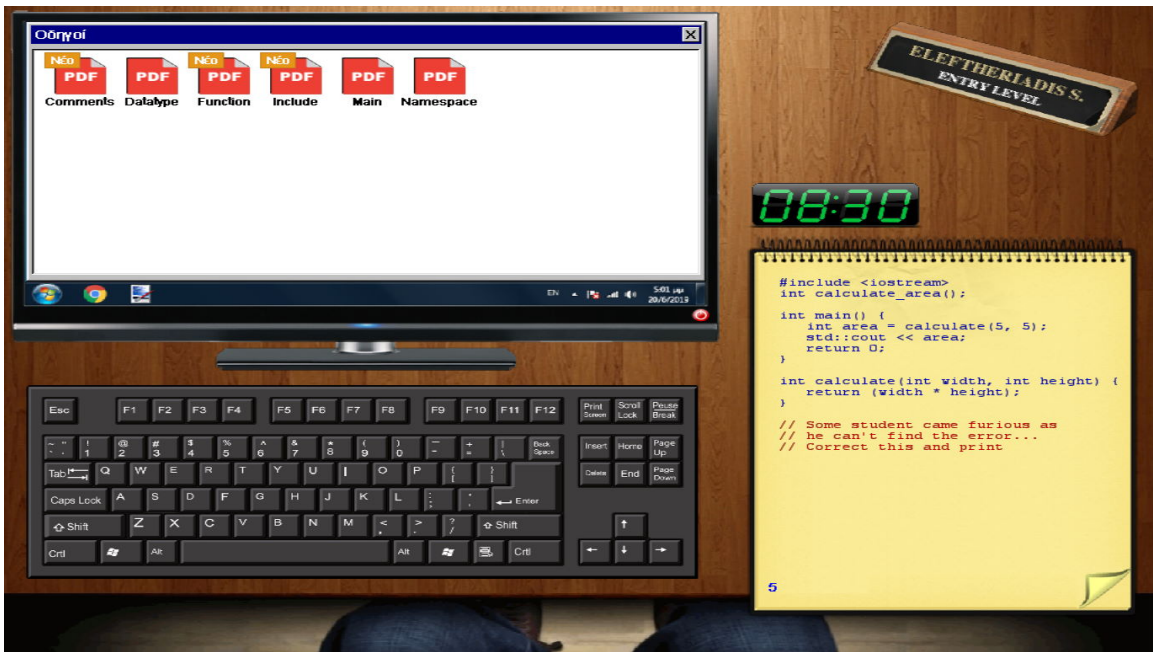
Εικόνα 17: Office Madness – Αξιολόγηση στον συντάκτη C++

5.5.4 Οδηγοί προγραμματιστικών εννοιών

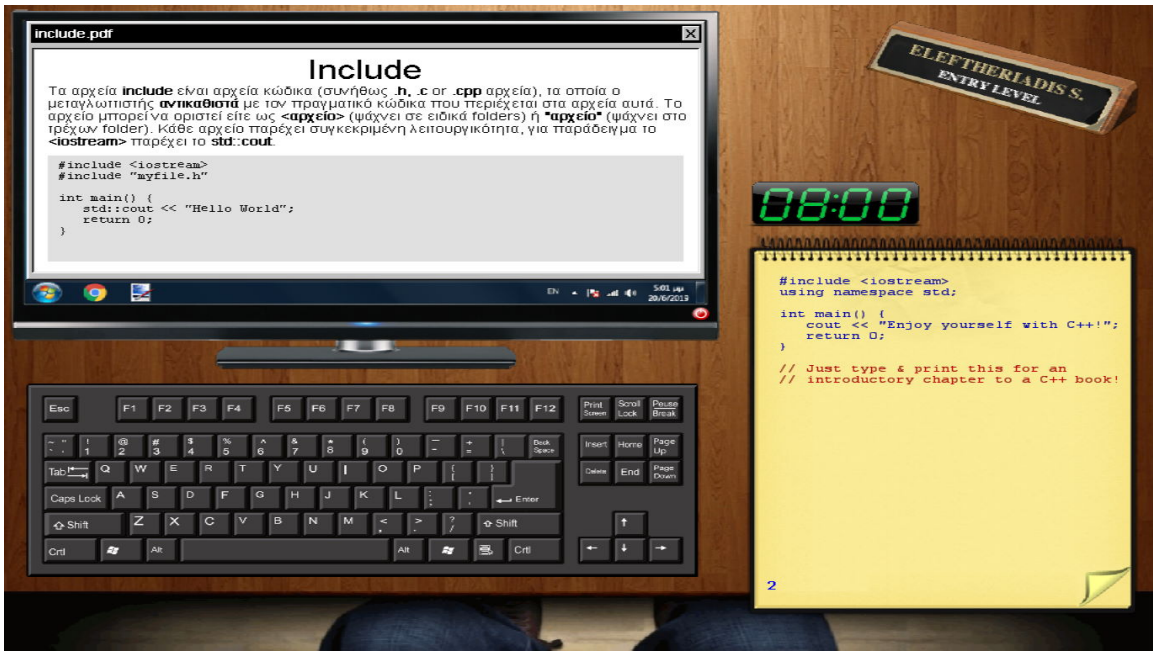
Πρόκειται για τη δεύτερη εφαρμογή της επιφάνειας εργασίας και δεν είναι παρά ένας κατάλογος (folder) ο οποίος γεμίζει, σταδιακά, με *εικονικά αρχεία PDF* τα οποία επεξηγούν, με αρκετή λεπτομέρεια, βασικές προγραμματιστικές έννοιες τις οποίες θα συναντήσει ο εικονικός εργαζόμενος κατά την εργασία του.

Τα αρχεία αυτά εισάγονται στην λίστα αυτή *αυτόματα*, όσο ο χρήστης μετακινείται μεταξύ των εργασιακών αναθέσεων του. Ρεαλιστικά, ο χρήστης θα έβρισκε τέτοιο υλικό στο Διαδίκτυο ή σε έντυπα βιβλία, οπότε η αυτόματη τοποθέτηση του υλικού στο παραπάνω folder είναι μια πρακτική σύμβαση που επιλέχθηκε για χάρη της γενικότερης απλοποίησης που υπάρχει στην εφαρμογή.

Τα αρχεία αυτά έχουν την γραφική μορφή των γνωστών εγγράφων *PDF* της Adobe ενώ έγινε προσπάθεια η μορφή στην οποία παρουσιάζεται το περιεχόμενο των οδηγών να είναι προσιτό, ευανάγνωστο και στοχευμένο ώστε να μην χωρίζεται η εφαρμογή σε εκπαίδευση ή σε εκτέλεση αλλά να συμβαίνουν και τα δυο (2) ταυτόχρονα, όπως θα συνέβαινε στην πραγματικότητα. Στην Εικόνα 18, παρουσιάζεται η λίστα ενός αριθμού οδηγών σε μορφή εικονικών αρχείων PDF. Στην Εικόνα 19 παρουσιάζεται η μορφή ενός επιλεγμένου οδηγού:



Εικόνα 18: Office Madness – Λίστα οδηγών προγραμματιστικών εννοιών



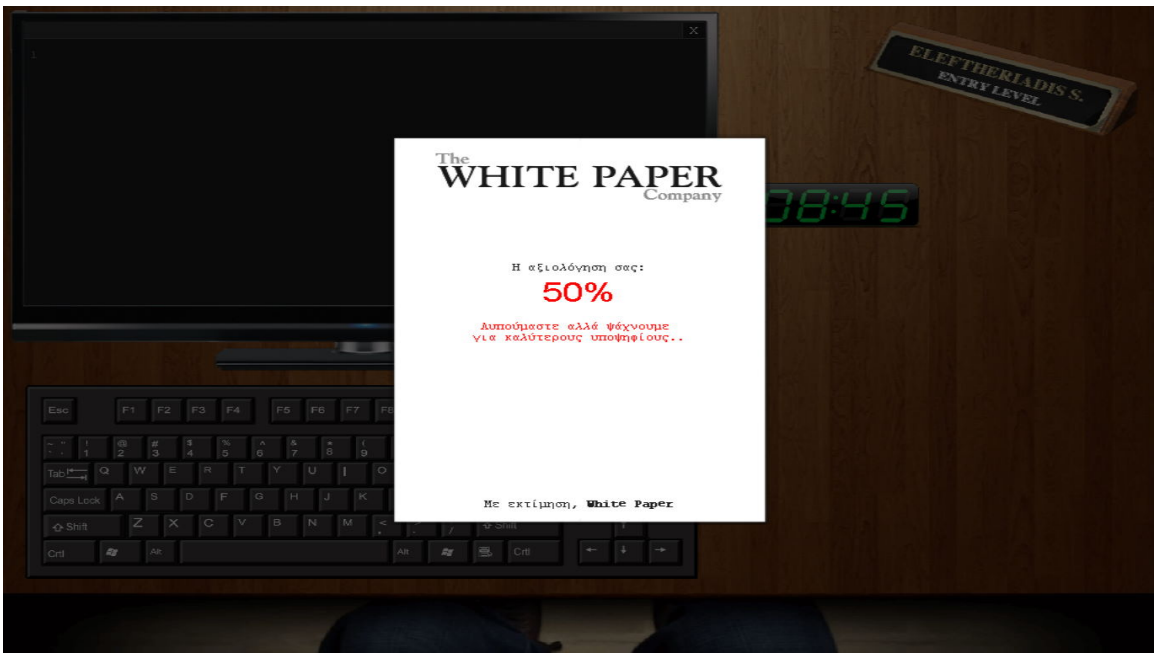
Εικόνα 19: Office Madness – Οδηγός προγραμματιστικής έννοιας

5.5.5 Φόρμα αξιολόγησης

Το παιχνίδι τελειώνει με δυο (2) τρόπους. Ο πρώτος είναι να ολοκληρώσει ο εικονικός εργαζόμενος τις προγραμματιστικές του αναθέσεις, δηλαδή να μεταγλωττίσει & να εκτυπώσει όλους τους διορθωμένους κώδικες για την θέση εργασίας που έχει αναλάβει π.χ. *Entry Level*. Ο δεύτερος τρόπος είναι μέσω της λήξης ωραρίου, η οποία συμβαίνει στις *15:30*, οπότε ο εργαζόμενος δεν μπορεί να κάνει κάτι παραπάνω.

Η αξιολόγηση της χρονικής περιόδου μιας (1) μέρας στην οποία κλήθηκε να δοκιμαστεί, εργασιακά, ο εικονικός εργαζόμενος έρχεται στην μορφή μιας φόρμας η οποία μας ενημερώνει, μέσω ενός ποσοστού πως τα πήγαμε και εάν καταφέραμε, τελικά, να προσληφθούμε στην εν λόγω εταιρεία. Με ένα υψηλό ποσοστό θα προσληφθούμε, ενώ με ένα μέτριο / κακό, θα απορριφθούμε.

Ωστόσο, το feedback που θα έχουμε αποκομίσει ακόμη και σε μια απόρριψη σίγουρα θα είναι θετικό, οπότε τις επόμενες φορές που θα δοκιμάσει ο χρήστης, είναι πιθανό να τα πάει καλύτερα, κάτι που είναι το τελικό ζητούμενο της εφαρμογής, συνεχής επανάληψη μιας διαδικασίας εκπαίδευσης του χρήστη που είναι σύντομη, έντονη & ρεαλιστική. Στην Εικόνα 20, παρουσιάζεται η αξιολόγηση απόδοσης, κατά την οποία ο χρήστης δυστυχώς *απορρίφθηκε*:



Εικόνα 20: Office Madness – Αξιολόγηση απόδοσης

5.6 Εκπαιδευτικό περιεχόμενο

Η εφαρμογή σοβαρού σκοπού ταξινομεί το εκπαιδευτικό περιεχόμενο σε δυο (2) εργασιακές θέσεις, μια εκ των οποίων επιλέγει ο χρήστης στην αρχή του παιχνιδιού. Οι θέσεις αυτές είναι οι Entry Level και Junior Developer.

Η θέση Entry Level είναι κατάλληλη για χρήστες που έχουν βασικές γνώσεις προγραμματισμού και οι ερωτήσεις που παρέχονται είναι κατάλληλα προσαρμοσμένες για αυτούς. Η θέση Junior Developer είναι κατάλληλη για πιο προχωρημένους χρήστες καθώς οι ερωτήσεις καταπιάνονται με δυσκολότερα θέματα της γλώσσας προγραμματισμού C++.

Ο χρήστης επιλέγει την εργασιακή θέση που επιθυμεί στην αρχή του παιχνιδιού, προσαρμόζοντας εμμέσως το *επίπεδο δυσκολίας* της εφαρμογής. Παρακάτω, παρουσιάζονται οι ερωτήσεις κάθε εργασιακής θέσης, το ζητούμενο κάθε προγραμματιστικής ανάθεσης καθώς και ο εκπαιδευτικός σκοπός της.

5.6.1 Προγραμματιστικές αναθέσεις

Το σύνολο προγραμματιστικών αναθέσεων για την εργασιακή θέση Entry Level είναι **24**, ενώ για την εργασιακή θέση Junior Developer είναι **14**. Κάθε ανάθεση παρουσιάζεται με έναν συγκεκριμένο τμήμα κώδικα, το οποίο έχει (συνήθως) κάποιο πρόβλημα, καθώς και ένα ζητούμενο, με τη μορφή *σχολίων* που υποθετικά συντάσσει η εταιρεία.

Ο χρήστης / εργαζόμενος πρέπει να συντάξει τον διορθωμένο κώδικα στον συντάκτη C++, διαβάζοντας προσεκτικά τα σχόλια της κάθε ανάθεσης. Στον Πίνακα 7, παρουσιάζεται το εκπαιδευτικό περιεχόμενο κάθε ανάθεσης της θέσης Entry Level, ενώ στον Πίνακα 8, παρουσιάζεται το εκπαιδευτικό περιεχόμενο της θέσης Junior Developer:

Πίνακας 7: Προγραμματιστικές αναθέσεις για Entry Level

Α/Α	Εκπαιδευτικό περιεχόμενο	Στόχος

1η	`int main() {`
	`  return 0;`
	`}`
	
	// We need some does-nothing
	// code for a beginner tutorial!
	// Please, compile & submit.
		Ο χρήστης πρέπει απλά να συντάξει τον κώδικα στον συντάκτη C++. Δεν υπάρχει κάποιο πρόβλημα στον κώδικα, ωστόσο βοηθάει τον χρήστη να κατανοήσει την **main** & να αρχίζει να διαβάζει τους οδηγούς.

2η	`#include <iostream>`
	
	`int main() {`
	`  std::cout << "Enjoy with C++!";`
	`  return 0;`
	`}`
	
	// Just type & submit this for an
	// introductory chapter to a C++ book!
		Στην εργασία αυτή, ο χρήστης μαθαίνει την λειτουργία των αρχείων **κεφαλίδων**, της **εξόδου κειμένου** και της χρήση του βασικού namespace **std::**.

3η	`#include <stdio.h>`
	`int main() {`
	`  int a = 5;`
	`  int b = (a * 2);`
	`  cout << b;`
	`  return 0;`
	`}`
	
	// A student complains about
	// some compiler errors..
	// Solve the errors, compile &
	// submit!
		Η εργασία αυτή αποσκοπεί στη χρήση της **σωστής κεφαλίδας** για χρήση της μεθόδου **cout**, καθώς και προσθήκης του namespace **std::**. Επίσης γίνεται πρώτη αναφορά στους τύπους δεδομένων όπως το **int,** όπου γίνεται εκτενής αναφορά στον αντίστοιχο οδηγό των **τύπων δεδομένων**, ο οποίος προστίθεται με την εργασία αυτή.

4η	`#include <stdio.h>`
	
	`int main() {`
	`  -- a comment line`
	`  /* a commented fragment`
	`  of code /*`
	`  std::printf("hello world");`
	`  return 0;`
	`}`
	
	// A student wants to comment
		Στην εργασία αυτή παρουσιάζονται οι **τύποι σχολίων**, τους οποίους ο χρήστης πρέπει να διαβάσει στον αντίστοιχο οδηγό & να διορθώσει στην εργασία. Επίσης, χρειάζεται να διορθώσει και πάλι το λανθασμένο **όνομα κεφαλίδας.** Η επαναλαμβανόμενη διόρθωση της σωστής κεφαλίδας θα ενισχύσει την μνήμη του χρήστη.

```
// some parts of his code.
// Solve his problems and submit!
```

5η

```
#include <iostream>
int calculate();

int main() {
   int area = calculate(5, 5);
   std::cout << area;
   return 0;
}

int calculate(int width, int height) {
   return (width * height);
}

// Some student came furious as
// he can't find the error...
// Correct this and submit.
```

Στην εργασία αυτή παρουσιάζεται η έννοια της **δήλωσης συνάρτησης** (declaration). Μέσω του αντίστοιχου οδηγού, ο χρήστης μαθαίνει την χρησιμότητα όχι μόνο της δήλωσης συνάρτησης, αλλά και της σωστής χρήσης **(προσθήκη παραμέτρων),** κάτι που τονίζεται και στην αξιολόγηση σε περίπτωση λάθους. Επίσης, γίνεται πρώτη αναφορά στην έννοια των **συναρτήσεων** και προστίθεται ο αντίστοιχος οδηγός με την εμφάνιση της εργασίας.

6η

```
#include <string>

namespace Me {
   int count = 0;
   std::string name;
   void set_name(std::string name) {
      Me::name = name;
   }
   std::string get_name() {
      return name;
   }
}

int main() {
   std::printf("Your name: ");
   std::printf("%s", Me::get_name());
   return 0;
}

// A student does not understand
// namespaces & their purpose.
// He wants to set the name 'Paul'
// and print the rest with 'printf'.
// Help him out, find the errors,
// compile & submit.
```

Ο χρήστης έρχεται σε πρώτη επαφή με τα **namespaces**, όπου μαθαίνει την χρησιμότητα τους μέσω του αντίστοιχου οδηγού. Σκοπός του είναι να ορίσει, με χρήση μεθόδων του namespace **Me**, ένα συγκεκριμένο όνομα και να το εκτυπώσει στην οθόνη. Τέλος, γίνεται μια μικρή αναφορά στην κεφαλίδα **string** όπου ο χρήστης αντιλαμβάνεται, μέσω του οδηγού, ότι πρόκειται για αντικείμενο που κρατάει ένα αλφαριθμητικό.

7η

```
int main() {
   cout << "Hello World";
   return 0;
}

// Some user wonders why he
// can't use 'cout'.
// Solve his errors & submit.
```

Στην εργασία αυτή, ο χρήστης θα κατανοήσει την σημασία της **εισαγωγής κεφαλίδας**. Στις προηγούμενες εργασίες διόρθωνε το όνομα και εδώ θα πρέπει να την εισάγει, όπως επίσης και το namespace **std::.** Όλα παρέχονται μέσω των οδηγών.

8η

```
#include <iostream>
#include <string>

int main() {
   string text = "Your name:";
   std::cout << text;
   return 0;
}

// A student is confused as
// he knows how to use 'string'.
// However, this gives compile errors.
// Find the errors, compile & submit.
```

Στόχος της εργασίας αυτής είναι να κατανοήσει ο χρήστης την σημασία του namespace **std::** και ότι θα πρέπει να εισάγεται είτε σε κάθε μέθοδο, είτε σε κάθε τύπο όπως το **string**.

9η

```
#include <iostream>

int main() {
   const int numbers[] = {1,2,3,4,5};
   std::cout << ...
   return 0;
}

// Some student wants to add the
// first and second number from
// array and print the result but
// he doesn't know how.
// Help him out, compile & submit.
```

Στην εργασία αυτή παρουσιάζεται η έννοια των **πινάκων** (arrays), καθώς και ο αντίστοιχος οδηγός που τα επεξηγεί. Στόχος του χρήστη είναι να χρησιμοποιήσει δυο (2) **στοιχεία του πίνακα** και να προσθέσει το περιεχόμενο τους. Τέλος, γίνεται πρώτη αναφορά στην έννοια της **σταθεράς,** όπου παρουσιάζεται μέσω του αντίστοιχου οδηγού.

10η

```
#include <cstdio>

int main() {
   float pi = 3.1416f;
   std::printf("2-decimal pi is = %f", pi);
   return 0;
```

Η εργασία αυτή κάνει χρήση της συνάρτησης εξόδου **printf(),** η οποία έχει παρουσιαστεί σε προηγούμενες εργασίες. Ωστόσο, ο χρήστης θα πρέπει να διαβάσει καλύτερα τον οδηγό **Output** και να

```
}

// A student wants control over
// his float number. However, his
// number is printed in whole while
// he only wants to print 2 decimals.
// Correct his code, compile & submit.
```

αντιληφθεί ποιο **specifier** της **printf()** πρέπει να χρησιμοποιήσει για να πετύχει τον στόχο του, δηλαδή της εκτύπωσης του PI, με χρήση μόνο **δυο (2) δεκαδικών ψηφίων**.

11η

```
#include <iostream>

int main() {
   double area, rad = 1.5;
   double pi = 3.141593;
   area = pi * rad * rad;
   std::cout << "Radius: " << rad;
   std::cout << ", Area: " << area;
   return 0;
}

// A student  wants to define a
// constant 'pi' to avoid updates
// on that variable, later.
// Correct his code & submit.
```

Στην εργασία αυτή, γίνεται σημαντικότερη χρήση των **σταθερών**, που αναφέρθηκαν σε προηγούμενη εργασία. Στόχος του χρήστη είναι να ορίσει την μεταβλητή pi ως **σταθερά** ώστε να μη μπορεί να τροποποιείται. Στην επαλήθευση ορθότητας του κώδικα, γίνεται έλεγχος εάν ο χρήστης έχει ορίσει **σωστά** την μεταβλητή του. Τέλος, ο χρήστης βλέπει και άλλους τύπους δεδομένων όπως είναι ο **double**.

12η

```
#include <sstream>

int main {
   cout << "Too hard to display",
        << " this text on my screen';
   return 0;
}

// A student has serious problems
// with basic C++ syntax. Correct
// his numerous errors, then
// compile & submit.
```

Στην εργασία αυτή, ο χρήστης έρχεται αντιμέτωπος με **άσχημα γραμμένο κώδικα**, σε μόλις δυο (2) γραμμές. Ο χρήστης θα πρέπει να **απλοποιήσει την διαδικασία**, να ορίσει δυο (2) μεθόδους **cout** ώστε να εκτυπώσει το κείμενο και να διορθώσει κάποια συντακτικά λάθη, τα οποία θα του επισημανθούν και από τον μεταγλωττιστή. Τέλος, θα πρέπει να διορθώσει (και πάλι) το λανθασμένο όνομα **κεφαλίδας**.

13η

```
#include <iostream>
#include <climits>

int main() {
   std::cout << "Ranges: ";
   std::cout << CHAR_MIN;
   std::cout << ", " << INT_MAX;
```

Ο χρήστης έρχεται αντιμέτωπος με κάποιες **σταθερές** οι οποίες υποδηλώνουν **όρια τιμών**. Μέσω της εξόδου αλλά και των οδηγών, αντιλαμβάνεται τη σημασία της ύπαρξης τους.

```
    std::cout << ", " << LONG_MAX;
    return 0;
}

// Just type & submit this for an
// introductory chapter to a C++ book!
```

14η

```
#include <cstdio>

int main() {
    int result = calculate_sum(5, 3);
    std::printf("result:  %d", result);
    return 0;
}

int calculate_sum(int a,int b) {
    return (a + b);
}

// A student complains about his
// code doesn't get compiled at all.
// Solve his errors, compile & submit.
```

Στην εργασία αυτή, ο χρήστης και πάλι θα πρέπει να προσθέσει μια **δήλωση συνάρτησης,** κάτι που ο χρήστης έχει συναντήσει και σε προηγούμενη εργασία. Στο τέλος, πρέπει να **εκτυπώσει** το αποτέλεσμα της συνάρτησης με τη **σωστή μέθοδο εξόδου**.

15η

```
#include <iostream>

int int_instruction_32() {
    return (1024 * 32);
}

int 586_cpu() {
    int num_cpu = 4;
}

int main() {
    if (int_instruction_32() > 256) {
        int ncpu = 586_cpu();
        std::cout << "ncpu = " << ncpu;
    }
    return 0;
}

// A student started writing a new
// compiler but his first version
// causes some errors.
// Solve his errors & tell him that
```

Στην, κάπως περίπλοκη, αυτή εργασία, ο χρήστης θα χρειαστεί να **διαβάσει** καλύτερα των **οδηγό των Data types** και να διορθώσει κάποια συντακτικά προβλήματα π.χ. όνομα συνάρτησης που **αρχίζει με αριθμό.** Θα πρέπει επίσης να **επιστρέψει σωστά** κάποιες μεταβλητές μέσω συναρτήσεων ώστε να παραχθεί η **σωστή έξοδος**.

```
// writing compilers isn't an easy
// job.
```

16η

```
#include <cstdio>

int main() {
   double x = 2;
   double y1, y2;
   y1 = std::pow("x", 3);
   y2 = std::pow(x + 4);
   std::printf("y1: %f", y1);
   std::printf(", y2: %f", y2);
   return 0;
}

// A student having problems with
// 'pow', not been able to use it.
// Help him out, compile & submit.
```

Στην εργασία αυτή παρουσιάζονται κάποιες νέες, **μαθηματικές συναρτήσεις**, που ανήκουν στην κεφαλίδα **<cmath>.** Ο χρήστης θα αντιληφθεί ότι το πρόγραμμα του δεν μεταγλωττίζει σωστά & μέσω αντίστοιχου οδηγού θα αντιληφθεί ποιο **αρχείο κεφαλίδας** θα πρέπει να εισάγει. Τέλος, θα μάθει **πως συντάσσονται,** με σωστό τρόπο, οι μαθηματικές συναρτήσεις που χρησιμοποιούνται εδώ.

17η

```
#include <iostream>
#include <cstdlib>

int main() {
   int r1, r2;
   std::srand(128);
   std::rand(r1);
   std::rand(r2);
   std::cout << "r1: " << r1;
   std::cout << ", r2: " << r2;
   return 0;
}

// A student can't compute a random
// number.
// Show him how, compile & submit.
```

Στην εργασία αυτή γίνεται πρώτη αναφορά στο **αρχείο κεφαλίδας <cstdlib>**, το οποίο μέσω αντίστοιχου οδηγού ο χρήστης μαθαίνει τον γενικότερο σκοπό του. Έχοντας διαβάσει τον οδηγό, θα είναι σε θέση να διορθώσει κατάλληλα τον προβληματικό κώδικα (**λάθος παράμετροι**).

18η

```
#include <iostream>

int main() {
   std::cout << "Sizes = ";
   std::cout << "c: " << sizeof(char);
   std::cout << ",i: " << sizeof(int);
   std::cout << ",s: " << sizeof(short);
   std::cout << ",l: " << sizeof(long);
   return 0;
```

Στη εργασία αυτή, ο χρήστης θα πρέπει αρχικά να διαβάσει τον οδηγό **sizeof** και να αντιληφθεί τον σκοπό του. Στη συνέχεια, απλώς θα συντάξει τον κώδικα που βλέπει. Δεν υπάρχει κάποιο πρόβλημα στον κώδικα.

```
}

// Just type & submit this for an
// introductory chapter to a C++ book!
```

19η

```
#include <cstdio>

int main() {
   int count = 1;
   std::printf("loop:");
   while(count <= 10) {
      std::printf(" %d", count);
      count++;
   }
   return 0;
}

// A c++ starter stumbled on a loop
// while trying to make it loop from
// 0 to 9, including the 9.
// Help him out with his loop problems.
```

Ο χρήστης έρχεται αντιμέτωπος, για πρώτη φορά, με **δομές επανάληψης** και συγκεκριμένα την **while**. Η εργασία τον ενημερώνει κατάλληλα ποιο είναι το ζητούμενο και θα πρέπει να **διορθώσει** κατάλληλα τον κώδικα ώστε να παράγει **σωστή έξοδο**.

20η

```
#include <cstdio>

int main() {
   int count = 0;
   std::printf("count:");
   while(count < 10) {
      std::printf(" %d",count);
      count++;
   }
   return 0;
}

// A student of c++ has been
// instructed to use the do-while
// to replicate the above code.
// Convert his code, compile & submit.
```

Σε συνέχεια της προηγούμενης εργασίας, ο χρήστης θα πρέπει να κάνει το ίδιο, χρησιμοποιώντας όμως αυτή τη φορά την δομή επανάληψης **do-while** για να εξάγει το ίδιο αποτέλεσμα**.**

21η

```
#include <iostream>

int main() {
   int key;
   key = 2;
   if (key == 1) {
```

Στην εργασία αυτή, ο χρήστης θα διαβάσει για τον έλεγχο συνθηκών μέσω της εντολής **if** και του αντίστοιχου οδηγού. Στη συνέχεια, θα πρέπει να διορθώσει κάποια συντακτικά προβλήματα.

```
            std::cout << "key is 1";
        else if (key == 2) {
            std::cout << "key is 2";
        }
        }
        else if (key == 3) {
            std::cout << "key is 3";
        }
        return 0;
    }

    // A student wants to prints some
    // text, based on 'key' value.
    // However, he somehow messed this up.
    // Teach him some proper if-else
    // practices, compile & submit.
```

22η

```
#include <iostream>

int main() {
    int number = 5;
    switch(5) {
        case number:
            std::cout << "number is 5";
        break;
        case number+1:
            std::cout << "number is 6";
        break;
        default:
            std::cout << "i don't know!";
    }
    return 0;
}

// A c++ user wants to check the
// value of a variable and print,
// based on that.
// However, his code has some errors.
// Fix them, compile & submit.
```

Στην εργασία αυτή παρουσιάζεται η έννοια του **switch**. Μέσω του αντίστοιχου οδηγού, ο χρήστης θα αντιληφθεί τη χρησιμότητα του και θα διορθώσει το προφανές λάθος του κώδικα ώστε να παραχθεί η σωστή έξοδος.

23η

```
#include <cstdio>

int main() {
    int x = 0;
    for(x=0;x<100;x++) {
```

Στην εργασία αυτή, γίνεται αναφορά στην δομή επανάληψης **for,** καθώς και στη χρησιμότητα του **break**. Ο χρήστης θα διαβάσει και για τις δυο (2) έννοιες στον

```
        ...
    }
    std::printf("current x = %d", x);
    return 0;
}

// A student wants to get out of a
// loop when the 'x' variable reaches
// 50 or more.
// Fill in the code, compile & submit.
```

αντίστοιχο οδηγό και θα είναι σε θέση να προσθέσει τον κατάλληλο κώδικα.

24η

```
#include <iostream>

struct Employee {
    int age;
};

int main() {
    Employee e;
    e.age = 41;
    ...

    std::cout << "name: " << ...;
    std::cout << ", surname: " << ...;
    std::cout << ", age: " << e.age;
    return 0;
}

// A student have been asked to
// complete a structure with a 'name'
// field with value 'Paul' and a
// 'surname' field with value 'Robins'.
// In addition, he asked to print all
// of them to screen.
// Fill the code, compile & submit.
```

Στην εργασία αυτή, παρουσιάζεται για πρώτη φορά η έννοια της δομής (**struct**). Μέσω του αντίστοιχου οδηγού, ο χρήστης θα καταλάβει τον σκοπό της δομής και πως θα μπορέσει να επιλύσει τα ζητούμενα της ανάθεσης.

Πίνακας 8: Προγραμματιστικές αναθέσεις για Junior Developer

Α/Α	Εκπαιδευτικό περιεχόμενο	Στόχος

1η	`#include <iostream>`
	`#include <string>`
	
	`int main() {`
	`  double x = 1.7;`
	`  int xref = &x;`
	`  xref += 1;`
	`  std::cout << "x: " << x;`
	`  std::cout << ", xref: " << xref;`
	
	`  const std::string s = "i am const";`
	`  std::string& const_s = s;`
	`  const_s = "..but need to update!";`
	`  std::cout << ", s: " << s;`
	`  return 0;`
	`}`
	
	`// This student has a hard time`
	`// understanding references.`
	`// Correct his code, compile & submit.`
		Στόχος του χρήστη στην εργασία αυτή είναι να κατανοήσει την έννοια των **αναφορών** (references), με βάση τον αντίστοιχο οδηγό. Θα πρέπει επίσης να διαβάσει περί **σταθερών** πάλι για να λύσει ένα δεύτερο πρόβλημα.

2η	`#include <iostream>`
	
	`int main() {`
	`  double x = 1;`
	`  double* xptr;`
	`  xptr = *x;`
	`  xptr = 2;`
	`  std::cout << "x: " << x;`
	`  return 0;`
	`}`
	
	// The same student came back again,
	// having a harder time understanding
	// pointers. He simply wants to print
	// the value of 'x' .
	// Correct his code, compile & submit.
		Στην εργασία αυτή, ο χρήστης έρχεται αντιμέτωπος για πρώτη φορά με **δείκτες** (pointers). Θα πρέπει να διαβάσει τον αντίστοιχο οδηγό, να τον κατανοήσει ώστε να λύσει κάποια συντακτικά προβλήματα και να εξάγει το σωστό αποτέλεσμα.

3η	`#include <cstdio>`
	`void swap(int*, int*);`
	
	`int main() {`
		Στην εργασία αυτή, ο χρήστης βλέπει μια συνάρτηση **swap** (μετάθεση τιμών), η οποία χρησιμοποιεί δείκτες. Θα πρέπει να την διορθώσει,

```
    int x = 10;
    int y = 20;
    swap(x, y);
    std::printf("x: %d, y: %d",x,y);
    return 0;
}

void swap(int* p1, int* p2) {
    int temp;
    temp = p1;
    p1 = p2;
    p2 = temp;
}

// Another user is having pointer
// problems, trying to develop a
// 'swap' function.
// Teach him good practices, compile
// and submit.
```

χρησιμοποιώντας τις γνώσεις του για τους **δείκτες**.

4η

```
#include <iostream>
void calc(int, int, int*);

int main() {
    int area = 0;
    calc(5, 5, area);
    std::cout << area;
    return 0;
}

void calc(int w, int h, int* area) {
    area = (w * h);
}

// A student has pointer problems.
// Help him out by correcting his
// compiling errors and submit.
```

Στην εργασία αυτή, ο χρήστης θα πρέπει να διορθώσει τον τρόπο με τον οποίο **καλείται μια συνάρτηση** που χρησιμοποιεί **δείκτες**, ώστε να **επιστρέψει** σωστά μια τιμή & να την εκτυπώσει.

5η

```
#include <iostream>

void get_sum(int a, int b, int sum) {
    sum = (a + b);
}

int main() {
```

Στην εργασία αυτή, ο χρήστης καλείται να χρησιμοποιήσει και πάλι **αναφορές** ώστε να διορθώσει μια συνάρτηση που **επιστρέφει** μια τιμή, μέσω **παραμέτρου** αναφοράς.

```
    int sum = 0;
    get_sum(2, 7, sum);
    std::cout << sum;
    return 0;
}

// This one complains that he is
// unable to get any sum using the
// 'get_sum' function. He was instructed
// that he should use a reference but
// he doesn't know how.
// Help him out and submit.
```

6η

```
#include <iostream>

int main() {
    const int w[] = {3,2,5,6,8};
    const int h[] = {4,2,6,3,5};
    int s = (sizeof(w) / sizeof(w[0]));
    for(int x=0;x<s;x++) {
        int a = (w[x] * h[x]);
        if (x > 0) {
            std::cout << ", ";
        }
        std::cout << "a: " << a;
    }
    return 0;
}

// A professor wants to share
// this code with his students.
// Just type, compile & submit.
```

Στην εργασία αυτή, ο χρήστης απλά καλείται να **συντάξει / αντιγράψει** ένα σχετικά περίπλοκο τμήμα κώδικα. Δεν υπάρχει κάποιο συντακτικό πρόβλημα.

7η

```
#include <cstdio>
#include <vector>

int main() {
    std::vector<int> numbers;
    numbers.push_back(1,2,3,4,5);
    for(int x=0;x<numbers.size;x++) {
        std::printf("%d ", numbers[x]);
    }
    return 0;
}
```

Στην εργασία αυτή, ο χρήστης μαθαίνει για πρώτη φορά για δομές δεδομένων και συγκεκριμένα για **vectors**. Μαθαίνει να χρησιμοποιεί τη **σωστή κεφαλίδα,** πως **προσθέτει** δεδομένα **κατάλληλου τύπου** σε μια δομή δεδομένων, αλλά και χρήσης τους σε δομή επανάληψης.

```
// A student wants to add some
// numbers in his vector of
// integers. However, his code
// causes compiling errors.
// Fix those errors and submit.
```

8η

```
#include <iostream>
#include <string>

int main() {
   std::string prompt = "Your name: ";
   std::string name('Peter');
   std::string line(2, '-');
   std::cout << line;
   std::cout << prompt << name;
   std::cout << line;
   return 0;
}

// A student almost succeeded using
// the 'string' class, having just
// an obvious error.
// Point his errors, compile & submit.
```

Στην εργασία αυτή, ο χρήστης έρχεται σε περισσότερο επαφή με την κλάση **string**. Διαβάζοντας καλύτερα τον οδηγό για τα Strings, θα κατανοήσει τους διαφορετικούς τρόπους που ορίζεται ένα string, ώστε να μπορέσει να διορθώσει κάποια συντακτικά προβλήματα ώστε να εξάγει το σωστό αποτέλεσμα.

9η

```
#if_def _MAIN_H_
#define _MAIN_H_

#define ONCE_DEFINED  "mytext"
#define PI            3.1415

#end_if

int main() {
   return 0;
}

// A progressing c++ student tried to
// use guards  (conditional inclusions)
// on a header file, but he constantly
// gets compiling errors.
// Fix his errors, compile & submit.
```

Στην εργασία αυτή, ο χρήστης μαθαίνει για **εντολές προεπεξεργαστή**. Θα πρέπει να διαβάσει τον αντίστοιχο οδηγό ώστε να αντιληφθεί πως ορίζονται ή διαγράφονται αλλά και πως χρησιμοποιούνται. Η εργασία αυτή έχει κάποια συντακτικά προβλήματα, που επιλύονται εύκολα με διάβασμα του οδηγού.

10η

```
#include <cstdio>

#define NEG(a,b)  ((a)<(b)?(a):(b))
```

Στην εργασία αυτή, απαιτούνται οι γνώσεις της προηγούμενης εργασίας για **εντολές προεπεξεργαστή**. Ο

```
#undefine NEG(a,b)
#define NEG(a, b)  ((a)<(b)?(-a):(-b))

int main() {
   int neg_v = NEG(1,3);
   std::printf("neg_v: %d", neg_v);
   return 0;
}

// The same c++ student tried to
// define & undefine a defined
// constant, but he gets compiling
// errors.
// Fix his errors, compile & submit.
```

χρήστης καλείται να **διαγράψει** μια **ορισμένη** σταθερά προεπεξεργαστή. Επίσης, ο χρήστης έχει μια πρώτη επαφή με **τριαδική συνθήκη** (ternary), για την οποία μπορεί να διαβάσει στον σχετικό οδηγό.

11η

```
#include <iostream>

#define PI   3.1415
...

int main() {
   std::cout << "PI: " << PI;
   std::cout << ", DBL_PI: " << ...;
   return 0;
}

// A student has been instructed to
// add an additional define (symbolic
// constant) to a program, a 'DBL_PI'.
// Fill in the code, compile & submit.
```

Στην εργασία αυτή, ο χρήστης θα πρέπει να εισάγει μια συγκεκριμένη **σταθερά προεπεξεργαστή** ώστε να εξάγει το σωστό αποτέλεσμα. Στόχος είναι να αντιληφθεί ο χρήστης τη **δυνατότητα υπολογισμών** στις σταθερές προεπεξεργαστή. Δίνονται ξεκάθαρα οι οδηγίες στο σχόλιο.

12η

```
#include <iostream>
#include <string>

int main() {
   std::string s1("Miss Winter");
   s1.insert(", Maria");
   std::cout << "s1: '" << s1 << "'";
   return 0;
}

// A c++ student made some beginner
// mistakes, using 'string' methods.
// He wanted to print the string
// 'Miss Winter, Maria'.
```

Στην εργασία αυτή, ο χρήστης καλείται να χρησιμοποιήσει και πάλι γνώσεις στα **strings**. Συγκεκριμένα, πρέπει να αντιληφθεί τις σωστές παραμέτρους μιας μεθόδου string (**insert**), ώστε να εξάγει το σωστό αποτέλεσμα.

```
// Fix his errors, compile & submit.
```

13η

```
#include <iostream>
#include <string>

int main() {
   std::string s1("There they go");
   int pos = s1.find("There");
   if (pos) {
      std::cout << "Found in: " << pos;
   }
   return 0;
}

// Intense 'string' work for the same
// c++ student and once again he's
// struggling. He expected to find
// some string and he didn't.
// Fix his errors, compile & submit.
```

Στην εργασία αυτή, ο χρήστης μαθαίνει **πως να ψάχνει** ένα τμήμα κειμένου σε ένα άλλο string, καθώς και σε ποια θέση. Ο οδηγός για **string** περιγράφει την δυνατότητα αυτή. Ωστόσο, η εργασία αυτή δεν έχει κάποιο συντακτικό πρόβλημα και ο χρήστης να πρέπει να σκεφτεί λίγο καλύτερα γιατί παίρνει λάθος αποτέλεσμα, συμβουλεύομενος πάντα τον οδηγό.

14η

```
#include <iostream>
#include <string>

class Account {
   private:
      std::string name;
   public:
      Account(std::string name) {
         this->name = "";
      }
      std::string get_name() {
         return "";
      }
};

int main() {
   Account acc;
   std::cout << acc.name;
   return 0;
}
// This student messed up his class.
// He asked to print the name 'Peter'
// by using the class methods.
// Fix his problems, compile & submit.
```

Στην εργασία αυτή, ο χρήστης έρχεται πρώτη φορά αντιμέτωπος με μια **κλάση** (class). Θα πρέπει να διορθώσει κάποια συντακτικά προβλήματα, ορίζοντας σωστά μεθόδους της κλάσης, καθώς και να μάθει τη σημασία της **κλήσης μεθόδων** αντί για τα **δεδομένα** της κλάσης.

6 Πιλοτική Αξιολόγηση του Παιχνιδιού

6.1 Εισαγωγή

Στο κεφάλαιο αυτό, πραγματοποιείται η πιλοτική αξιολόγηση της εφαρμογής σοβαρού σκοπού που αναπτύχθηκε στα πλαίσια της διπλωματικής εργασίας. Για την πιλοτική αξιολόγηση, θα αξιοποιηθούν οι απαντήσεις του ερωτηματολογίου, το οποίο δημιουργήθηκε για τις ανάγκες της αξιολόγησης και δημοσιεύθηκε στο Διαδίκτυο:

Διεύθυνση ερωτηματολογίου:
https://docs.google.com/forms/d/1smKvZhJyFN-TPhF43zQmzWP4tB8rW_hd-8j7TqxJdyI

6.2 Δομή ερωτηματολογίου

Το ερωτηματολόγιο είναι βασισμένο στο πλαίσιο αξιολόγησης MEEGA (Savi, R., Wangenheim, G. C., & Borgato, F. A., 2011) και αφορά ερωτήσεις οι οποίες έχουν ταξινομηθεί σε ένα σύνολο *παραγόντων ποιότητας* & *διαστάσεων*, από τους δημιουργούς του πλαισίου αξιολόγησης. Κάθε ερώτηση ανήκει σε μια *διάσταση* και κάθε διάσταση ανήκει σε έναν *παράγοντα ποιότητας*.

Το MEEGA είναι ένα πλαίσιο αξιολόγησης, το οποίο σχεδιάστηκε ειδικά για την αξιολόγηση εκπαιδευτικών παιχνιδιών. Παρέχει γρήγορη αξιολόγηση με χρήση ειδικών ερωτηματολογίων ενώ δεν απαιτεί εξειδικευμένες γνώσεις περί εκπαιδευτικής θεωρίας ή στατιστικής από όσους το υιοθετούν. Εφαρμόζεται μέσω ενός κατάλληλα δομημένου ερωτηματολογίου για τη συλλογή δεδομένων από τους χρήστες ενός εκπαιδευτικού παιχνιδιού και έχει ως στόχο την αξιολόγηση εκπαιδευτικών παιχνιδιών με βάση το *Κίνητρο*, την *Εμπειρία* και τη *Μάθηση*. Στον Πίνακα 9 παρουσιάζονται οι *παράγοντες ποιότητας*:

Πίνακας 9: Ερωτηματολόγιο - Παράγοντες ποιότητας MEEGA

Παράγοντες ποιότητας
Κίνητρο *(Motivation)*
Εμπειρία χρήστη *(User experience)*
Μάθηση *(Learning)*

Οι παραπάνω *παράγοντες ποιότητας* ορίζουν επίσης και ένα σύνολο *διαστάσεων,* στις οποίες ταξινομείται κάθε ερώτηση του ερωτηματολογίου. Κάθε *διάσταση* ανήκει σε έναν *παράγοντα ποιότητας* και μπορεί να περιέχει μια ή παραπάνω ερωτήσεις. Ο Πίνακας 10 παρουσιάζει τις *διαστάσεις* αυτές, καθώς και σε ποιον *παράγοντα ποιότητας* ανήκουν:

Πίνακας 10: Ερωτηματολόγιο – Διαστάσεις MEEGA

Διάσταση	**Παράγοντας ποιότητας**
Προσοχή *(Attention)*	Κίνητρο
Συνάφεια *(Relevance)*	Κίνητρο
Αυτοπεποίθηση *(Confidence)*	Κίνητρο
Ικανοποίηση *(Satisfaction)*	Κίνητρο
Εμβύθηση *(Immersion)*	Εμπειρία χρήστη
Κοινωνική αλληλεπίδραση *(Social Interaction)*	Εμπειρία χρήστη
Πρόκληση *(Challenge)*	Εμπειρία χρήστη
Διασκέδαση *(Fun)*	Εμπειρία χρήστη
Ικανότητα *(Competence)*	Εμπειρία χρήστη
Ψηφιακό παιχνίδι *(Digital Game)*	Εμπειρία χρήστη
Βραχυπρόθεσμη μάθηση *(Short-term learning)*	Μάθηση
Μακροπρόθεσμη μάθηση *(Long-term learning)*	Μάθηση

Ο παράγοντας ποιότητας *κίνητρο* βασίζεται στο μοντέλο *A.R.C.S.* (Keller, 1987), ένα γνωστό μοντέλο το οποίο έχει χρησιμοποιηθεί σε αρκετές μελέτες για να αξιολογήσει το κίνητρο των μαθητών, αξιοποιώντας μεταξύ άλλων και τα εκπαιδευτικά παιχνίδια.

Ο παράγοντας ποιότητας *εμπειρία χρήστη* χρησιμοποιεί την εμβύθηση *(immersion)*, την έντονη δηλαδή εμπλοκή του παίχτη στο παιχνίδι, την *πρόκληση*, την *διασκέδαση* και την *ικανότητα* ως τα κύρια χαρακτηριστικά του ή *διαστάσεις.* Η διάσταση *κοινωνική αλληλεπίδραση*, καθώς και οι ερωτήσεις της, δεν αξιοποιήθηκε, καθώς το παιχνίδι δεν υποστηρίζει αλληλεπίδραση χρηστών.

Ο παράγοντας ποιότητας *μάθηση* χωρίζεται σε βραχυπρόθεσμη μάθηση, οι ερωτήσεις της οποίας βασίζονται σε πιο άμεσους εκπαιδευτικούς στόχους, ενώ η μακροπρόθεσμη μάθηση αφορά ερωτήσεις που έχουν αντίκτυπο στην επαγγελματική ενασχόληση του χρήστη με το αντικείμενο.

Κάθε μια από τις παραπάνω *διαστάσεις* αφορούν ουσιαστικά ιδιότητες του παιχνιδιού, οι οποίες αξιολογούνται από τις απαντήσεις των ερωτήσεων. Στον Πίνακα 11, παρουσιάζεται το σύνολο των *ερωτήσεων*, καθώς και η *διάσταση* και ο *παράγοντας ποιότητας* στα οποία ανήκουν. Όπως αναφέρθηκε και παραπάνω, οι ερωτήσεις της διάστασης *κοινωνικής αλληλεπίδρασης* δεν αξιοποιήθηκαν στο ερωτηματολόγιο, λόγω απουσίας αλληλεπίδρασης μεταξύ χρηστών στην εφαρμογή, ωστόσο οι ερωτήσεις συμπεριλήφθηκαν παρακάτω. Οι απαντήσεις που παρέχονται είναι πάντα διαβαθμισμένες, συνήθως από το *Διαφωνώ απόλυτα* (ή καθόλου) μέχρι το *Συμφωνώ απόλυτα,* κατά το σύστημα likert με πενταβάθμια κλίμακα (1 = Διαφωνώ απόλυτα, 5 = Συμφωνώ απόλυτα).

Πίνακας 11: Ερωτηματολόγιο – Ερωτήσεις MEEGA

Προσοχή (Κίνητρο)
1.1. Ο σχεδιασμός του παιχνιδιού είναι ελκυστικός.
1.2. Υπήρχε κάτι ενδιαφέρον στην αρχή του παιχνιδιού που τράβηξε την προσοχή μου.
1.3. Η ποικιλομορφία του παιχνιδιού (φόρμα, περιεχόμενο ή δραστηριότητες) με βοήθησε να διατηρήσω την προσοχή μου στο παιχνίδι.
Συνάφεια (Κίνητρο)
2.1. Το περιεχόμενο του παιχνιδιού σχετίζεται με τα ενδιαφέροντά μου.
2.2. Ο τρόπος λειτουργίας του παιχνιδιού ταιριάζει στον τρόπο μάθησης μου.
2.3. Το περιεχόμενο του παιχνιδιού συνδέεται με άλλες γνώσεις που είχα ήδη.
Αυτοπεποίθηση (Κίνητρο)
3.1. Ήταν εύκολο να καταλάβω το παιχνίδι και να αρχίσω να το χρησιμοποιώ ως υλικό μελέτης.
3.2. Περνώντας από το παιχνίδι, ένιωθα σίγουρος ότι μάθαινα.
Ικανοποίηση (Κίνητρο)
4.1. Είμαι ικανοποιημένος γιατί ξέρω ότι θα έχω ευκαιρίες να χρησιμοποιώ στην πράξη. πράγματα που έμαθα παίζοντας αυτό το παιχνίδι.
4.2. Λόγω της προσωπικής μου προσπάθειας κατάφερα να προχωρήσω στο παιχνίδι.
Εμβύθηση (Εμπειρία χρήστη)
5.1. Προσωρινά, ξέχασα την καθημερινότητα μου. Έχω επικεντρωθεί πλήρως στο παιχνίδι.
5.2. Δεν παρατήρησα το πέρασμα του χρόνου, ενώ έπαιζα. Όταν το παρατήρησα, το παιχνίδι είχε ήδη τελειώσει.
5.3. Ένιωσα τον εαυτό μου περισσότερο μέσα στο παιχνίδι παρά στην πραγματική ζωή, ξεχνώντας τι ήταν γύρω μου.
Κοινωνική αλληλεπίδραση (Social interaction)
6.1. Κατάφερα να αλληλεπιδράσω με άλλους κατά τη διάρκεια του παιχνιδιού

6.2. Διασκέδασα με τους άλλους παίκτες
6.3. Το παιχνίδι προωθεί τη συνεργασία και / ή τον ανταγωνισμό μεταξύ των παικτών
Πρόκληση (Εμπειρία χρήστη)
7.1. Αυτό το παιχνίδι είναι κατάλληλο για μένα, οι δραστηριότητες δεν είναι ούτε πολύ εύκολες ούτε πολύ δύσκολες.
7.2. Το παιχνίδι εξελίσσεται με καλό ρυθμό και δεν γίνεται μονότονο - προσφέρει νέες προκλήσεις, καταστάσεις ή παραλλαγές στις δραστηριότητες του.
7.3. Τι ποσοστό επιτυχίας ολοκλήρωσης των εργασιών έφτασα κατά τη χρήση της εφαρμογής.
7.4. Για ποια θέση εργασίας κάνατε αίτηση μέσα στην εφαρμογή;
Διασκέδαση (Εμπειρία χρήστη)
8.1. Διασκέδασα με το παιχνίδι.
8.2. Θα συνιστούσα αυτό το παιχνίδι στους συναδέλφους μου.
8.3. Θα ήθελα να παίξω ξανά αυτό το παιχνίδι.
Ικανότητα (Εμπειρία χρήστη)
9.1. Πέτυχα τους στόχους του παιχνιδιού, εφαρμόζοντας τις γνώσεις μου.
9.2. Είχα θετικά συναισθήματα για την αποτελεσματικότητα αυτού του παιχνιδιού.
Ψηφιακό παιχνίδι (Εμπειρία χρήστη)
10.1. Ο χειρισμός της εκτέλεσης ενεργειών στο παιχνίδι είχε καλή απόκριση.
10.2. Είναι εύκολο να μάθω πώς να χρησιμοποιώ τη διεπαφή και τον χειρισμό του παιχνιδιού.
Βραχυπρόθεση μάθηση (Μάθηση)
11.1. Το παιχνίδι συνέβαλε στη εκπαίδευση μου σε αυτό το αντικείμενο.
11.2. Το παιχνίδι ήταν αποτελεσματικό για τη εκπαίδευση μου, συγκρίνοντάς το με άλλες δραστηριότητες του μαθήματος.
11.3. Ο χρόνος που έπαιξα.
Μακροπρόθεσμη μάθηση (Μάθηση)
12.1. Η εμπειρία με το παιχνίδι θα συμβάλει στην επαγγελματική μου απόδοση στην πράξη.

Εκτός από τις ερωτήσεις εκπαιδευτικού περιεχομένου, οι χρήστες έπρεπε να απαντήσουν σε ένα σύνολο γενικών ερωτήσεων, οι οποίες παρουσιάζονται στον Πίνακα 12:

Πίνακας 12: Ερωτηματολόγιο - Γενικές ερωτήσεις

Γενικές ερωτήσεις
Μορφωτικό επίπεδο
Ηλικιακή ομάδα
Πόσο συχνά παίζετε ηλεκτρονικά παιχνίδια
Πόσο συχνά παίζετε μη-ηλεκτρονικά παιχνίδια (κάρτες ή επιτραπέζια παιχνίδια κλπ.)
Σε τι επίπεδο γνωρίζετε μια αντικειμενοστραφή γλώσσα προγραμματισμού (C++, Java)

6.3 Αποτελέσματα

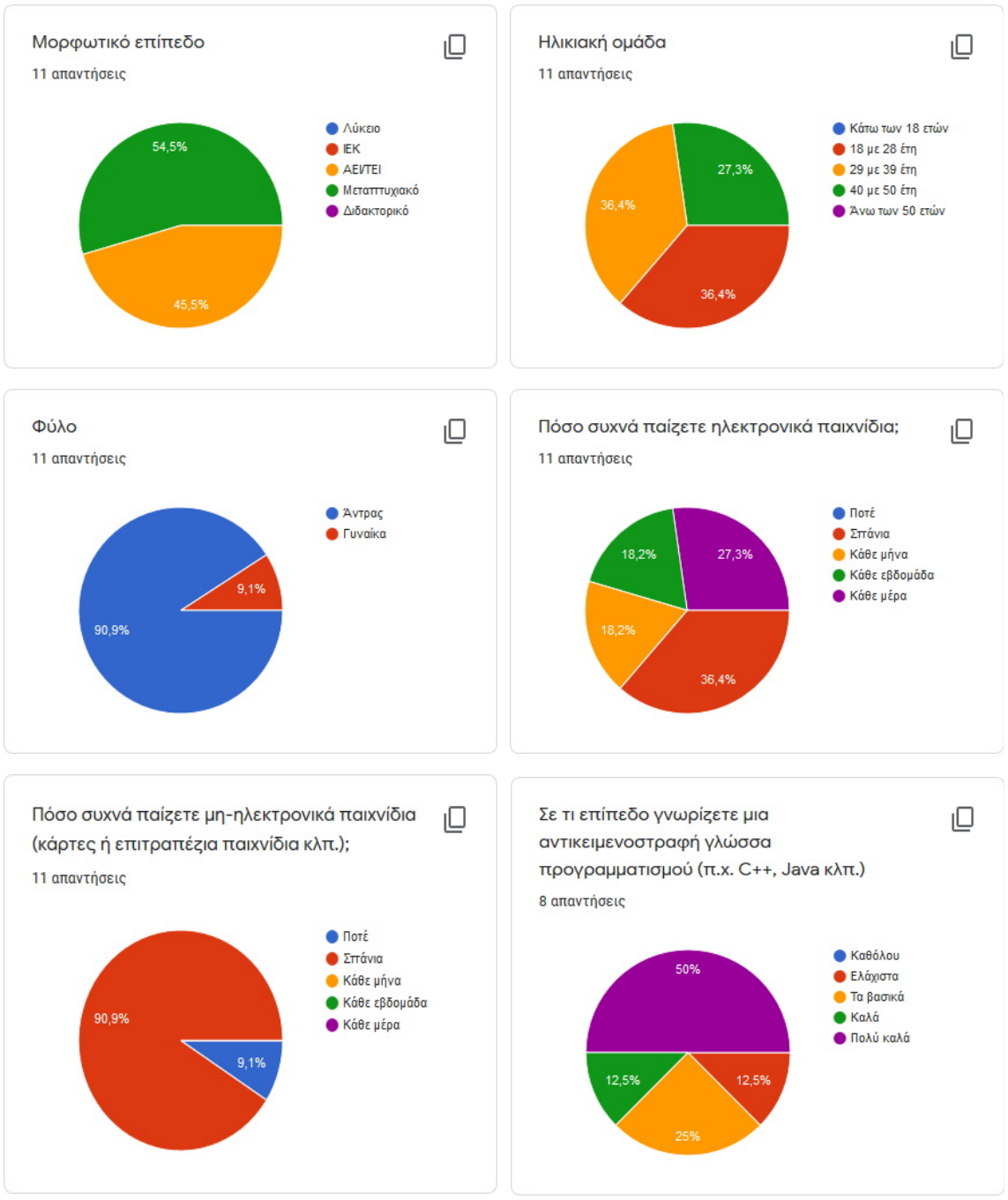
Την εφαρμογή αξιολόγησαν *11 χρήστες*, σε ένα χρονικό διάστημα *10 ημερών*. Τα αποτελέσματα της πιλοτικής αξιολόγησης παρουσιάζονται παρακάτω, συγκεντρωτικά, ανά κατηγορία.

Η πρώτη κατηγορία ερωτήσεων, οι γενικές ερωτήσεις, αφορούν δημογραφικά στοιχεία των χρηστών. Οι επόμενες κατηγορίες αφορούν ερωτήσεις που περιέχονται στις *διαστάσεις* του ερωτηματολογίου MEEGA.

Τα αποτελέσματα της πιλοτικής αξιολόγησης χρησιμοποιούν τον ενσωματωμένο μηχανισμό των *Google Forms* για να παραχθούν και παρουσιάζονται σε μορφή στατιστικών γραφημάτων.

6.3.1 Γενικές ερωτήσεις

Στην Εικόνα 21, παρουσιάζονται, συγκεντρωτικά, τα αποτελέσματα των *γενικών ερωτήσεων* του ερωτηματολογίου*:*



Εικόνα 21: Αποτελέσματα ερωτηματολογίου – Γενικές ερωτήσεις

Σύμφωνα με τα παραπάνω αποτελέσματα, το *μορφωτικό επίπεδο* των συμμετεχόντων χρηστών που αξιολόγησαν είναι υψηλό (45.5% AEI/TEI, 54.5% Μεταπτυχιακό), κάτι που ήταν αναμενόμενο καθώς υποδεικνύει την άμεση σύνδεση μεταξύ μορφωτικού επιπέδου & εκπαίδευσης με ηλεκτρονικά μέσα.

Οι χρήστες *18 με 28 έτη* και *29 με 39 έτη* (72.8%) αποτελούν το μεγαλύτερο ποσοστό των *ηλικιακών ομάδων* που απασχολήθηκαν στο παιχνίδι. Οι ηλικίες αυτές, επίσης, αποτελούν την πλειοψηφία του ενεργού εργατικού δυναμικού του τομέα της Πληροφορικής (Stack Overflow Developer Survey Results, 2019) οπότε τα αποτελέσματα της έρευνας θα αντικατοπτρίζουν και την πραγματικότητα.

Όσο αφορά το *Φύλο*, ήταν αναμενόμενο η συντριπτική πλειοψηφία να αποτελείται από άντρες (90.9%), ωστόσο στον τομέα της πληροφορικής δεν παίζει ιδιαίτερο ρόλο αυτό. Πολλές γυναίκες απασχολούνται πλέον στον χώρο της Πληροφορικής, ωστόσο στον ειδικό τομέα του *προγραμματισμού* είναι δύσκολο ακόμη να υπάρχει ισορροπία στο φύλο.

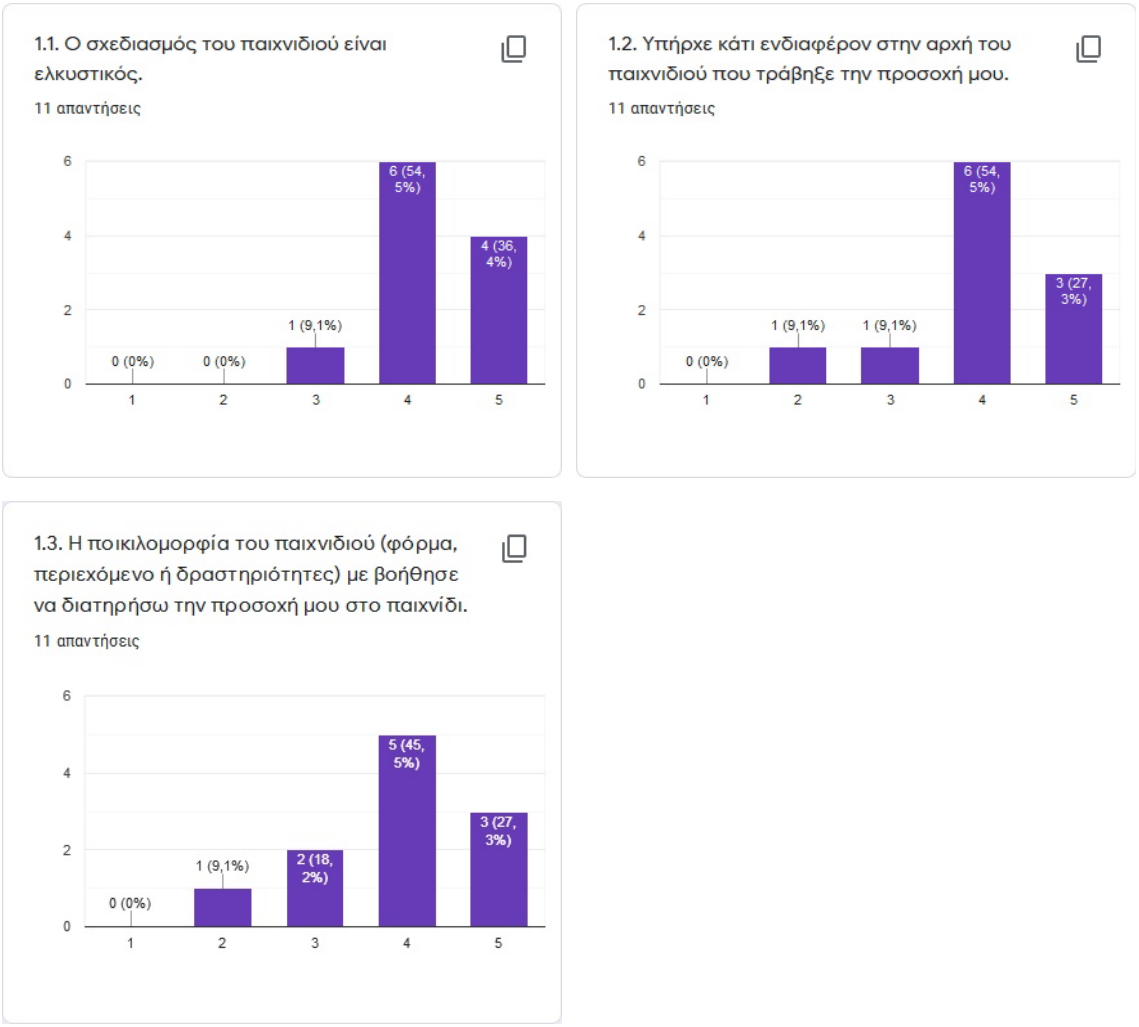
Η ερώτηση που είχε ενδιαφέρον αφορούσε τη *χρήση ηλεκτρονικών παιχνιδιών*, της οποίας τα αποτελέσματα ποικίλουν. Καταρχήν, όλοι οι συμμετέχοντες παίζουν παιχνίδια (100%), κάτι που υποδεικνύει μια γενικότερη τάση. Ωστόσο, αρκετοί είναι αυτοί που παίζουν είτε σπάνια (36.4%) είτε κάθε μέρα (27%), κάτι που δείχνει επίσης μια γενικότερη διακύμανση στους χρήστες.

Μια άλλη ερώτηση, της οποία τα αποτελέσματα προκάλεσαν ενδιαφέρον ήταν η *χρήση μη-ηλεκτρονικών παιχνιδιών*. Ενδεχομένως να αφορά και την γενικότερη τάση της κοινωνίας να απομακρύνεται από τα συμβατικά επιτραπέζια παιχνίδια, με κάρτες ή χωρίς, και να προτιμάει τις ηλεκτρονικές εκδόσεις αντίστοιχων παιχνιδιών. Το μεγαλύτερο ποσοστό παίζει σπάνια πλέον (90.9%) τα κλασσικά επιτραπέζια παιχνίδια, ενώ το υπόλοιπο 9.1% δεν παίζει καθόλου.

Τέλος, η ερώτηση περί *γνώσης κάποιας αντικειμενοστρεφούς γλώσσας προγραμματισμού* από τους χρήστες, έφερε μη αναμενόμενα αποτελέσματα. Η πλειοψηφία των χρηστών (50%) γνωρίζει τουλάχιστον μια αντικειμενοστρεφή γλώσσα προγραμματισμού σε πολύ καλό βαθμό, κάτι που δεν ήταν αναμενόμενο, τουλάχιστον σε τέτοιο βαθμό και ποσοστό χρηστών. Εξαιρετικά ενδιαφέρον το γεγονός ότι όλοι οι συμμετέχοντες έχουν κάποια γνώση αντικειμενοστρεφούς γλώσσας προγραμματισμού, έστω και ελάχιστη (12.5%).

6.3.2 Προσοχή (Attention)

Στην Εικόνα 22, παρουσιάζονται, συγκεντρωτικά, τα αποτελέσματα των ερωτήσεων του ερωτηματολογίου που αφορούν την διάσταση *Προσοχή* του ερωτηματολογίου MEEGA.



Εικόνα 22: Αποτελέσματα ερωτηματολογίου – Προσοχή

Ο *σχεδιασμός του παιχνιδιού* αποδείχθηκε *ελκυστικός* για τους περισσότερους χρήστες, με το 36.4% να συμφωνεί απόλυτα. Σε αυτό υπάρχει η πεποίθηση ότι βοήθησε ένα σύνολο από σχεδιαστικούς παράγοντες, όπως οι συγκεκριμένοι στόχοι και διαδικασίες, το ρεαλιστικό περιβάλλον, οι απλοί & διακριτοί μηχανισμοί καθώς και ο εύκολος χειρισμός.

Το παιχνίδι προκάλεσε μεγάλο *ενδιαφέρον* από την *αρχή της χρήσης* του, σύμφωνα με την πλειοψηφία των συμμετεχόντων (81.8%). Σε αυτό υπάρχει η πεποίθηση ότι βοήθησε αρκετά η ελκυστική εισαγωγή του παιχνιδιού που

προϊδέαζε τον χρήστη, καθώς και το γενικότερο περιβάλλον του παιχνιδιού το οποίο είναι έξω από τα συνηθισμένα.

Τέλος, οι *δραστηριότητες,* το *περιεχόμενο* και η *μορφή του παιχνιδιού* αποδείχθηκε ικανότατος παράγοντας να διατηρήσει την προσοχή των χρηστών (72.8%), ωστόσο υπάρχουν περιθώρια βελτίωσης, σύμφωνα με κάποιους χρήστες (18.2%), οι οποίοι ωστόσο δεν ήταν αρνητικοί, απλά είχαν ουδέτερη στάση.

6.3.3 Συνάφεια (Relevance)

Στην Εικόνα 23, παρουσιάζονται, συγκεντρωτικά, τα αποτελέσματα των ερωτήσεων του ερωτηματολογίου που αφορούν τη διάσταση *Συνάφεια* του ερωτηματολογίου MEEGA.

Εικόνα 23: Αποτελέσματα ερωτηματολογίου – Συνάφεια

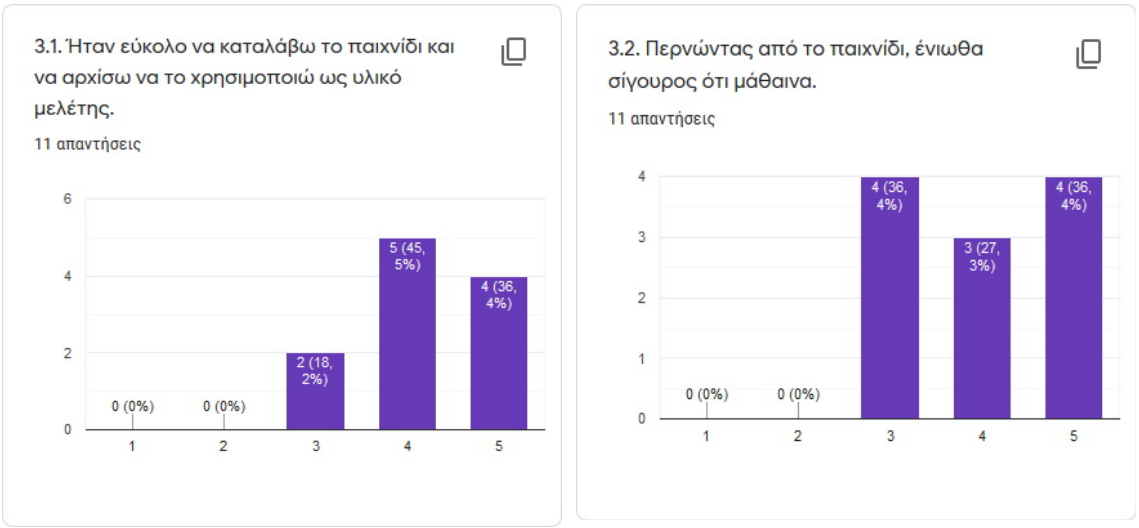
Όσον αφορά τη *συνάφεια* του παιχνιδιού, το μεγαλύτερο ποσοστό των συμμετεχόντων (91%), *συσχέτισε τα ενδιαφέροντα του με το εκπαιδευτικό περιεχόμενο* του παιχνιδιού. Υπάρχει η πεποίθηση ότι η παρουσίαση του εκπαιδευτικού περιεχομένου, όπως ο κώδικας και οι οδηγοί χρήσης, ήταν κάτι που αναμενόταν από τους συμμετέχοντες χρήστες.

Ο *τρόπος λειτουργίας* του παιχνιδιού προκάλεσε κάποια προβλήματα στον τρόπο μάθησης των συμμετεχόντων, οι οποίοι πιθανότατα έχουν συνηθίσει σε ευκολότερους & πιο συμβατικούς τρόπους μάθησης. Ένας σημαντικός αριθμός χρηστών (45.5%), ανταπεξήλθε στις δυσκολίες ενός νέου τρόπου λειτουργίας, όμως αποδείχθηκε ότι υπάρχουν πολλά περιθώρια βελτίωσης σύμφωνα με ένα 45.5%, οι οποίοι είχαν ουδέτερη στάση.

Τέλος, οι χρήστες έμειναν ικανοποιημένοι (90.9%) από τη *συνάφεια περιεχομένου*, καθώς αυτό *συσχετιζόταν με γνώσεις που είχαν ήδη,* κάτι που βοηθούσε στην μεταφορά της νέας γνώσης. Υπάρχει η πεποίθηση ότι αρκετοί χρήστες βοηθήθηκαν αρκετά από τη συνεχή επανάληψη κώδικα που παρουσιάζεται, κάτι που βελτίωνε τη διαδικασία μάθησης τους.

6.3.4 Αυτοπεποίθηση (Confidence)

Στην Εικόνα 24, παρουσιάζονται, συγκεντρωτικά, τα αποτελέσματα των ερωτήσεων του ερωτηματολογίου που αφορούν την διάσταση *Αυτοπεποίθηση* του ερωτηματολογίου MEEGA.



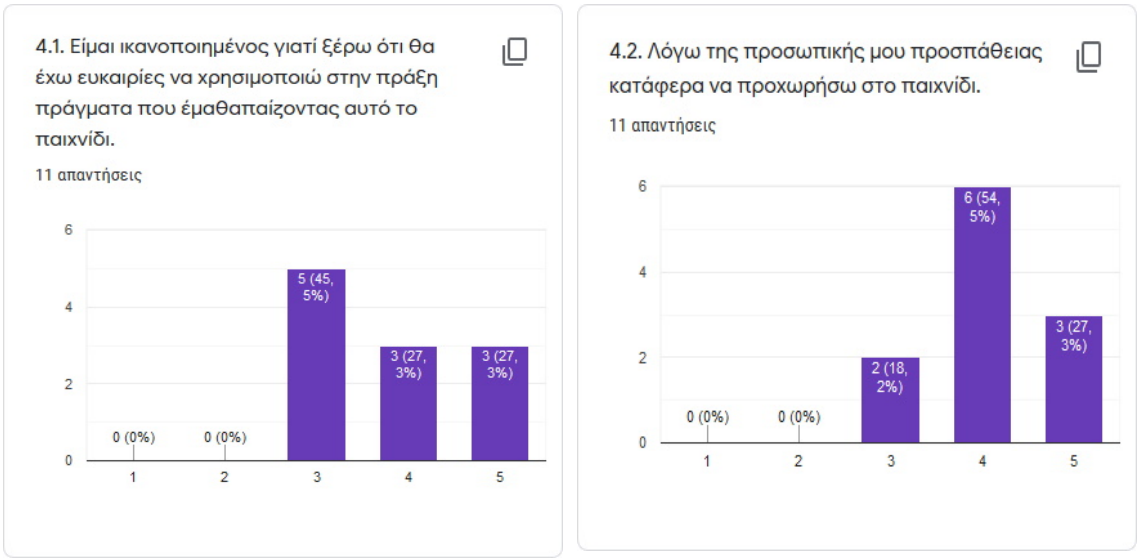
Εικόνα 24: Αποτελέσματα ερωτηματολογίου – Αυτοπεποίθηση

Το παιχνίδι αποδείχθηκε ευκολότερο στην κατανόηση, απ' ότι ήταν αναμενόμενο, από τους χρήστες, κάτι που συνέβαλε στην ευκολία *χρήσης του ως υλικό μελέτης* από την πλειοψηφία των χρηστών (81.9%).

Ωστόσο, ποικίλα αποτελέσματα έδωσε η ικανότητα του παιχνιδιού να *μεταδώσει γνώση κατά τη χρήση* του από τους συμμετέχοντες. Οι περισσότεροι χρήστες (63.7%) αντιλήφθηκαν ότι μαθαίνουν μέσω του παιχνιδιού, ωστόσο ένα σημαντικό 36.4% των χρηστών είχαν ουδέτερη στάση. Το αποτέλεσμα αυτό δείχνει έναν τομέα του παιχνιδιού που πρέπει να βελτιωθεί, αυτόν της σύνδεσης μάθησης & αυτοπεποίθησης.

6.3.5 Ικανοποίηση (Satisfaction)

Στην Εικόνα 25, παρουσιάζονται, συγκεντρωτικά, τα αποτελέσματα των ερωτήσεων του ερωτηματολογίου που αφορούν τη διάσταση *Ικανοποίηση* του ερωτηματολογίου MEEGA.



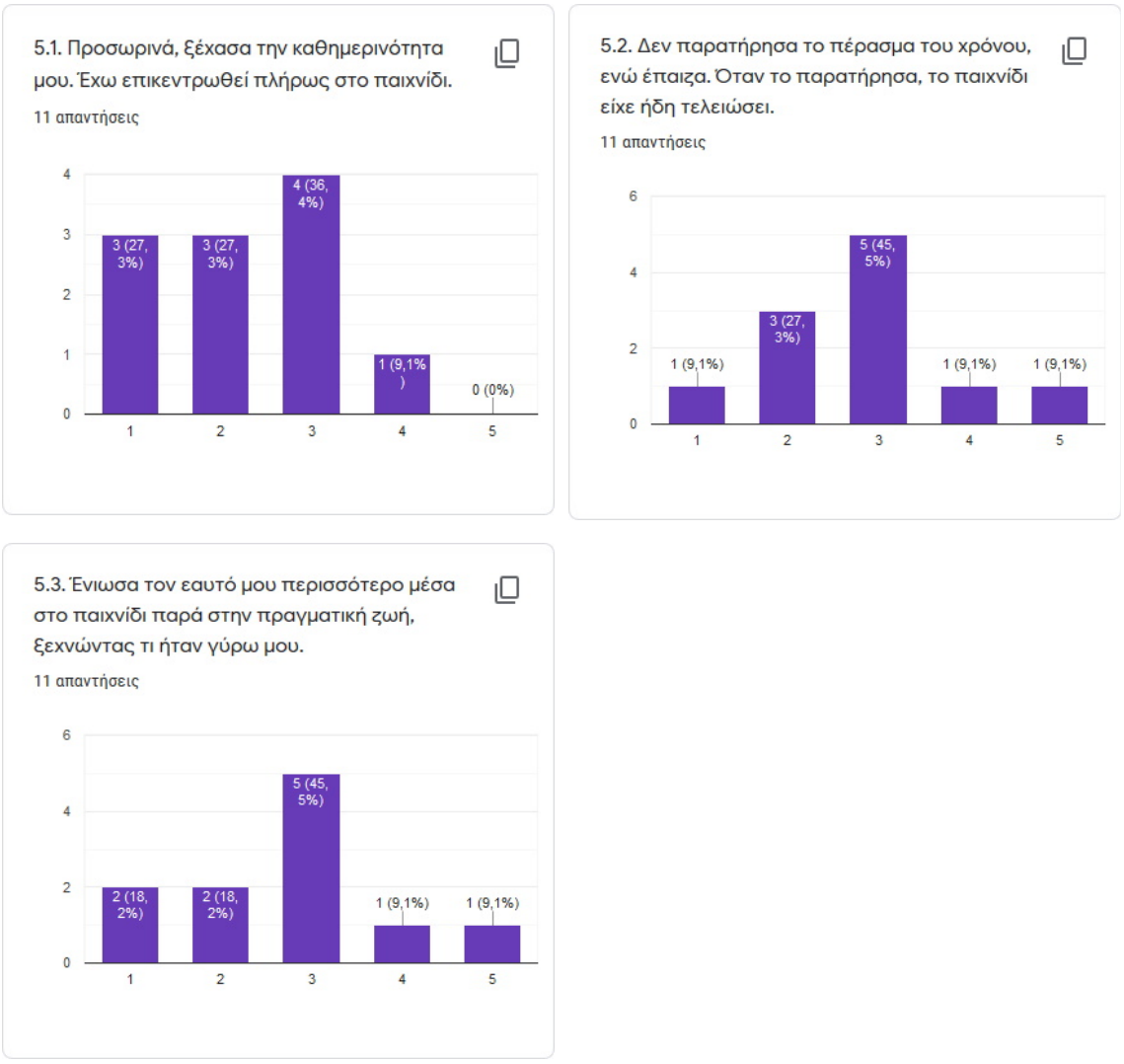
Εικόνα 25: Αποτελέσματα ερωτηματολογίου – Ικανοποίηση

Όσον αφορά την *ικανοποίηση* από τη χρήση του παιχνιδιού και της αξιοποίησης των γνώσεων αυτών στην πράξη, τα αποτελέσματα ποικίλουν. Πολλοί χρήστες (54.6%) ένιωσαν ικανοποιημένοι ότι η γνώση τους θα αξιοποιηθεί και στην πράξη, ωστόσο αρκετοί χρήστες (45.5%) εκφράζουν μια ουδέτερη στάση. Υπάρχει μεγάλο περιθώριο βελτίωσης στον τομέα αυτό.

Η πλειοψηφία των συμμετεχόντων (81.8%) αισθάνθηκε ικανοποιημένη που μπόρεσε να *προχωρήσει μέσα στο παιχνίδι*, αξιοποιώντας τις γνώσεις που αποκόμισε από τη χρήση του. Ωστόσο, ένα σημαντικό 18.2% είχε ουδέτερη στάση. Ένα βασικό συμπέρασμα είναι το εκπαιδευτικό περιεχόμενο να παρουσιάζεται περισσότερο ολοκληρωμένο ώστε ο χρήστης να αισθάνεται πιο έτοιμος & σίγουρος για τις γνώσεις του.

6.3.6 Εμβύθηση (Immersion)

Στην Εικόνα 26, παρουσιάζονται, συγκεντρωτικά, τα αποτελέσματα των ερωτήσεων του ερωτηματολογίου που αφορούν τη διάσταση *Εμβύθηση* του ερωτηματολογίου MEEGA.



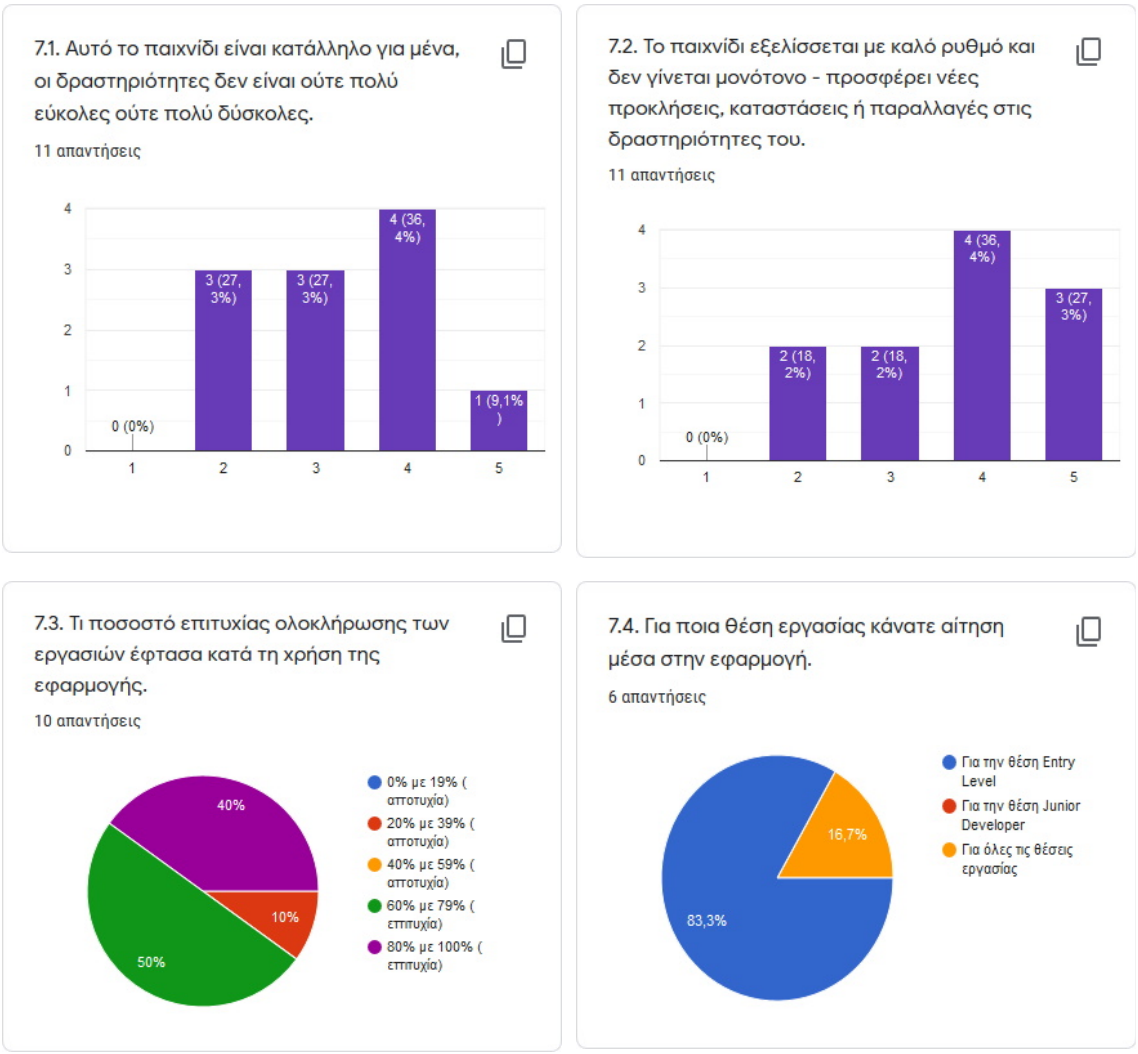
Εικόνα 26: Αποτελέσματα ερωτηματολογίου – Εμβύθηση

Στον τομέα της εμβύθησης (immersion), το παιχνίδι δεν πέτυχε καλή επίδοση. Συγκεκριμένα, αρκετοί χρήστες (54.6%) δεν ένιωσαν *να επικεντρώνονται ιδιαίτερα στο παιχνίδι* και να *ξεχνάνε την καθημερινότητα* τους. Αρκετοί χρήστες (36.4%) επίσης είχαν μια ουδέτερη στάση. Το παιχνίδι δεν τους ταξίδεψε, με επιτυχία, σε ένα εικονικό γραφείο. Υπάρχει η πεποίθηση ότι το επαναλαμβανόμενο μοτίβο του παιχνιδιού, τα λιγοστά εφέ & η συνεχής εργασία κούρασε τους παίχτες κατά τη χρήση, είναι κάτι όμως που μπορεί να βελτιωθεί.

Το *πέρασμα του χρόνου* γίνεται αντιληπτό από αρκετούς χρήστες (36.4%), κάτι που δεν είναι ιδιαίτερα θετικό και συνδυάζεται με την προηγούμενη ερώτηση. Ένα μεγάλο ποσοστό (45.5%) επίσης είχε μια ουδέτερη στάση. Το βασικό συμπέρασμα είναι ότι χρειάζονται προσθήκες στο παιχνίδι που θα το ξεμπλοκάρουν από μονότονες διαδικασίες & θα ενισχύσουν την εμβύθηση.

6.3.7 Πρόκληση (Challenge)

Στην Εικόνα 27, παρουσιάζονται, συγκεντρωτικά, τα αποτελέσματα των ερωτήσεων του ερωτηματολογίου που αφορούν τη διάσταση *Πρόκληση* του ερωτηματολογίου MEEGA.



Εικόνα 27: Αποτελέσματα ερωτηματολογίου – Πρόκληση

Στο αίσθημα της *πρόκλησης*, το παιχνίδι τα κατάφερε σχετικά καλά. Οι *δραστηριότητες* & το *εκπαιδευτικό περιεχόμενο* του παιχνιδιού φαίνεται να βρίσκεται στο σωστό επίπεδο για αρκετούς χρήστες (45.5%), ούτε πολύ δύσκολες ούτε πολύ εύκολες. Ωστόσο, αρκετοί χρήστες έχουν ουδέτερη στάση (27.3%), ενώ ένα 27.3% διαφωνεί. Το συμπέρασμα είναι ότι θα πρέπει τόσο οι δραστηριότητες, όσο και το εκπαιδευτικό περιεχόμενο να προσαρμοστεί κατάλληλα ώστε να υπάρχει ομαλότερη μετάβαση από το εύκολο στο δύσκολο.

Το παιχνίδι κατάφερε να *παρέχει καλό ρυθμό και μην γίνεται μονότονο κατά τη χρήση του,* σύμφωνα με τους περισσότερους συμμετέχοντες (63.7%). Ωστόσο, κάποιοι χρήστες είτε είχαν ουδέτερη στάση (18.2%), είτε θεώρησαν ότι το παιχνίδι γίνεται μονότονο (18.2%), πιθανότατα ύστερα από αρκετή χρήση. Ένα βασικό συμπέρασμα είναι ότι, όπως και στην *εμβύθηση*, θα πρέπει να γίνει προσθήκη νέων ή τροποποίηση δραστηριοτήτων, μειώνοντας την πιθανότητα μονοτονίας.

Το *ποσοστό ολοκλήρωσης των προγραμματιστικών αναθέσεων* κατά την τελική αξιολόγηση από την εφαρμογή, ήταν θετικό. Το σύνολο σχεδόν των χρηστών (90%) πέτυχε θετική αξιολόγηση στο *60%* και πάνω, επιλύοντας τις περισσότερες προγραμματιστικές αναθέσεις, ενώ αρκετοί συμμετέχοντες (40%) επίλυσαν το *80%* και πάνω.

Τέλος, οι περισσότεροι συμμετέχοντες (83.3%) επέλεξαν τη θέση Entry Level, πιθανότατα για να ελέγξουν το ευκολότερο επίπεδο του παιχνιδιού πρώτα. Ωστόσο, δεν υπήρξε ιδιαίτερη συμμετοχή (16.7%) στο επίπεδο Junior Developer. Ενδεχομένως, αυτό να σημαίνει ότι το Entry Level δεν είναι και τόσο εύκολο ή είναι αρκετά μεγάλο, κάτι που κουράζει τον χρήστη και δεν τον οδηγεί εύκολα στην επιλογή του δυσκολότερου επιπέδου.

6.3.8 Διασκέδαση (Fun)

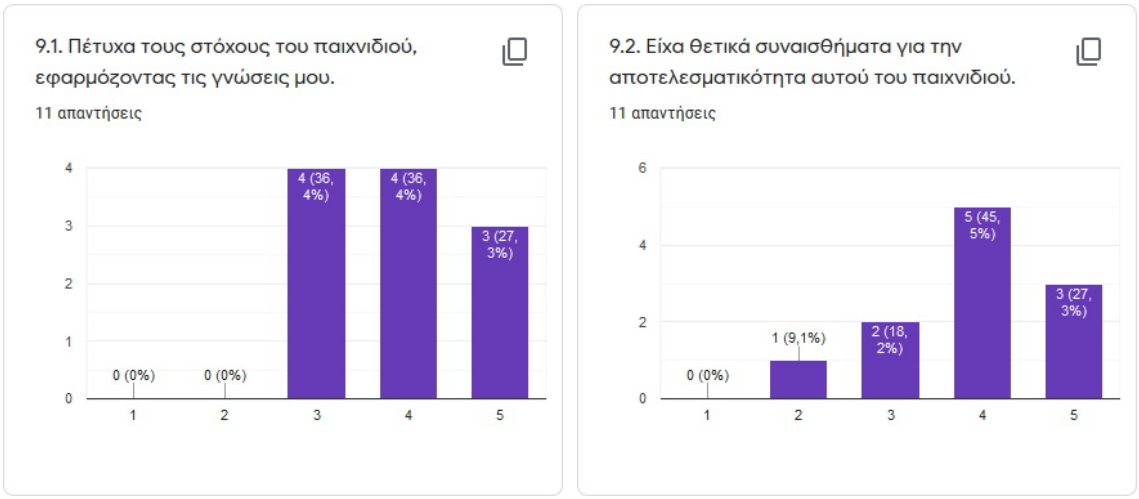
Στην Εικόνα 28, παρουσιάζονται, συγκεντρωτικά, τα αποτελέσματα των ερωτήσεων του ερωτηματολογίου που αφορούν τη διάσταση *Διασκέδαση* του ερωτηματολογίου MEEGA.

Εικόνα 28: Αποτελέσματα ερωτηματολογίου – Διασκέδαση

Οι περισσότεροι συμμετέχοντες (72.8%) φαίνεται να *διασκέδασαν με το παιχνίδι,* σύμφωνα με τα αποτελέσματα. Αυτό ενισχύεται από τη γενικότερη διάθεση των χρηστών να *συνιστούν το παιχνίδι* στους συναδέλφους τους, όπως και το ότι θα *ήθελαν να το ξαναπαίξουν*. Το παιχνίδι φαίνεται να τα κατάφερε πολύ καλά στον τομέα της *διασκέδασης,* κάτι που δεν ήταν αναμενόμενο, σε τέτοιο βαθμό, λόγω της ιδιαίτερης μορφής της εφαρμογής.

6.3.9 Ικανότητα (Competence)

Στην Εικόνα 29, παρουσιάζονται, συγκεντρωτικά, τα αποτελέσματα των ερωτήσεων του ερωτηματολογίου που αφορούν τη διάσταση *Ικανότητα* του ερωτηματολογίου MEEGA.



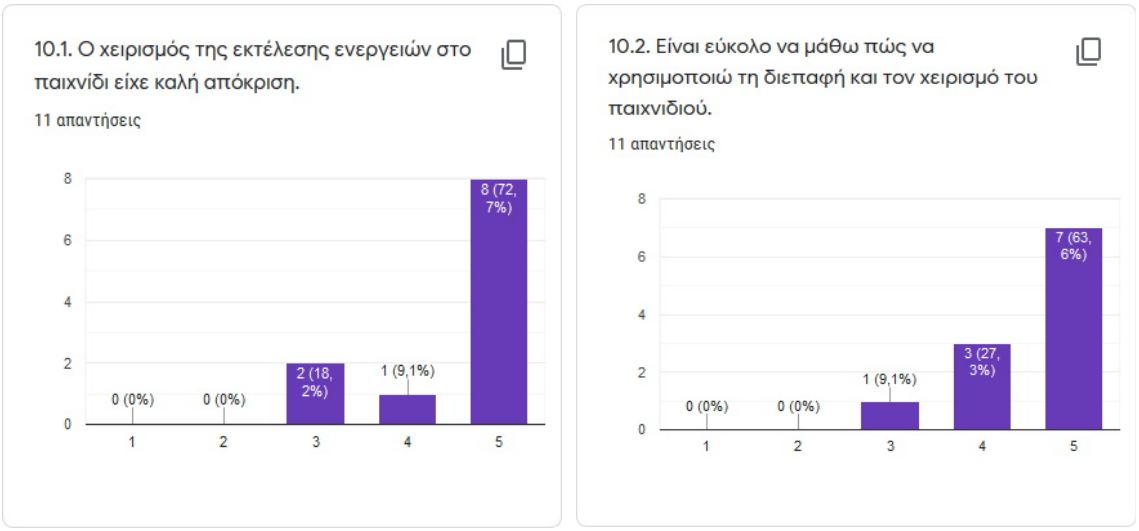
Εικόνα 29: Αποτελέσματα ερωτηματολογίου – Ικανότητα

Η πλειοψηφία των χρηστών (63.7%) φαίνεται να *πέτυχε τους στόχους του παιχνιδιού, εφαρμόζοντας τις γνώσεις* που αποκόμισαν από το παιχνίδι, κάτι που συνέβη και στην αξιολόγηση της *ικανοποίησης*. Ωστόσο, αρκετοί χρήστες έχουν ουδέτερη στάση (36.4%), κάτι που χρήζει βελτίωσης.

Οι περισσότεροι χρήστες (72.8%) έχουν *θετικά συναισθήματα από την αποτελεσματικότητα του παιχνιδιού*. Ωστόσο, όπως και προηγουμένως, υπάρχουν και κάποιοι χρήστες που παραμένουν ουδέτεροι (18.2%). Το βασικό συμπέρασμα είναι ότι θα πρέπει είτε να βελτιωθούν οι οδηγοί προγραμματιστικών εννοιών, είτε να απλοποιηθούν οι προγραμματιστικές αναθέσεις, κάτι που θα βελτιώσει το *επίπεδο γνώσεων* των συμμετεχόντων, όσο και το *αίσθημα αποτελεσματικότητας της εφαρμογής*.

6.3.10 Ψηφιακό παιχνίδι (Digital Game)

Στην Εικόνα 30, παρουσιάζονται, συγκεντρωτικά, τα αποτελέσματα των ερωτήσεων του ερωτηματολογίου που αφορούν τη διάσταση *Ψηφιακό παιχνίδι* του ερωτηματολογίου MEEGA.



Εικόνα 30: Αποτελέσματα ερωτηματολογίου – Ψηφιακό παιχνίδι

Ο *χειρισμός του παιχνιδιού* αποδείχτηκε εξαιρετικά επιτυχημένος, σύμφωνα με το σύνολο σχεδόν των συμμετεχόντων (81.8%), κάτι που συνέβη και με την *εκμάθηση της διεπαφής* που χρησιμοποιείται στο παιχνίδι. Ωστόσο, ένα σημαντικό 18.2% είχε ουδέτερη στάση. Υπάρχει η πεποίθηση ότι η προσθήκη του ρεαλιστικού στοιχείου στο παιχνίδι, τόσο σε γραφικό όσο και λειτουργικό επίπεδο, βοήθησε σε μεγάλο βαθμό στην κατανόηση του χειρισμού.

6.3.11 Βραχυπρόθεσμη μάθηση (Short-term learning)

Στην Εικόνα 31, παρουσιάζονται, συγκεντρωτικά, τα αποτελέσματα των ερωτήσεων του ερωτηματολογίου που αφορούν τη διάσταση *Βραχυπρόθεσμη μάθηση* του ερωτηματολογίου MEEGA.

Εικόνα 31: Αποτελέσματα ερωτηματολογίου – Βραχυπρόθεσμη μάθηση

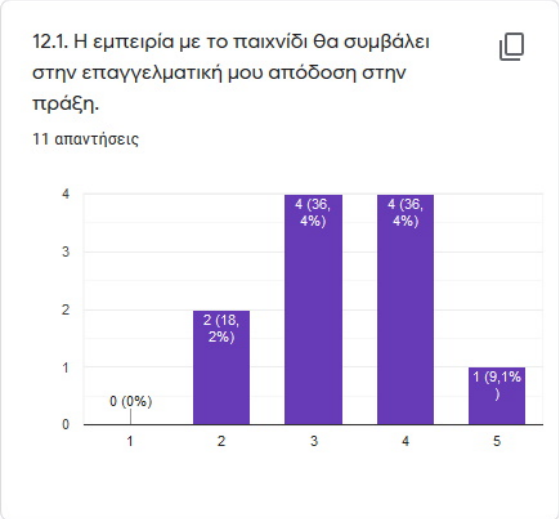
Η χρήση του παιχνιδιού φαίνεται ότι κατάφερε να *συμβάλλει στην εκπαίδευση*, για έναν μεγάλο αριθμό συμμετεχόντων (54.6%). Ωστόσο, αρκετοί συμμετέχοντες (36.4%) έχουν ουδέτερη στάση. Το βασικό συμπέρασμα είναι ότι γράφτηκε και στην διάσταση της *ικανοποίησης*. Το εκπαιδευτικό περιεχόμενο πρέπει να παρουσιάζεται περισσότερο ολοκληρωμένο ώστε ο χρήστης να αισθάνεται πιο έτοιμος & σίγουρος για τις γνώσεις του.

Όσον αφορά τον χρόνο χρήσης της εφαρμογής, τα αποτελέσματα ποικίλουν. Οι περισσότεροι συμμετέχοντες (70%) ασχολήθηκαν περίπου 30’

λεπτά με μια (1) ώρα, χρόνος ικανός για εξαγωγή αποτελεσμάτων & εκμάθησης. Ωστόσο, ένα πολύ θετικό στοιχείο της αξιολόγησης αφορούσε την χρήση της εφαρμογής άνω των δυο (2) ωρών από ένα 20% των χρηστών.

6.3.12 Μακροπρόθεσμη μάθηση (Long-term learning)

Στην Εικόνα 32, παρουσιάζονται, συγκεντρωτικά, τα αποτελέσματα των ερωτήσεων του ερωτηματολογίου που αφορούν την διάσταση *Μακροπρόθεσμη μάθηση* του ερωτηματολογίου MEEGA.



Εικόνα 32: Αποτελέσματα ερωτηματολογίου – Μακροπρόθεσμη μάθηση

Όσον αφορά την *μακροπρόθεσμη μάθηση*, τα αποτελέσματα της αξιολόγησης ποικίλουν. Αρκετοί συμμετέχοντες (45.5%) βοηθήθηκαν αρκετά από το παρεχόμενο εκπαιδευτικό περιεχόμενο, ενώ αρκετοί (36.4%) εκφράζουν μια ουδέτερη στάση. Το βασικό συμπέρασμα είναι ότι θα πρέπει να γίνουν τροποποιήσεις, κυρίως στις άλλες *διαστάσεις* που προηγήθηκαν, ώστε να βελτιωθεί σταδιακά και η *διάσταση* αυτή.

6.3.13 Σχόλια

Κατά την αξιολόγηση της εφαρμογής, οι συμμετέχοντας έγραψαν κάποιες παρατηρήσεις & προτάσεις βελτίωσης, οι οποίες παρουσιάζονται παρακάτω:

- “Πολύ καλή προσπάθεια με πολλά θετικά στοιχεία. Τα γραφικά του παιχνιδιού ήταν ενδιαφέροντα και η ροή του ομαλή χωρίς προβλήματα. Προς βελτίωση: 1) Εφόσον ο παίκτης μαθαίνει μελετώντας τους οδηγούς, ο χρόνος πρέπει να σταματά κατά τη μελέτη τους. 2) Ίσως το επίπεδο "Entry", πρέπει να γίνει ευκολότερο ή να δωθεί περισσότερος χρόνος. 3) Επειδή το σκηνικό που διαδραματίζεται το παιχνίδι είναι το Γραφείο, πρέπει όλες οι φράσεις ‘ένας μαθητής/φοιτητής..’ να αντικατασταθούν με ‘ένας συνάδελφος/ο προϊστάμενος...’.”

- “Σαν πρόταση βελτίωσης θα μπορούσε να μπει η χρήση του TAB στον editor, καθώς η έλλειψη του είναι λίγο αρνητική“

- “Signup form:
  The Name field would more clearly be named the First Name field, to balance out the Last Name field. It was not clear to me what the difference is between Entry level and Junior developer, as in practice, I believe in the industry juniors are considered entry level.
  Code editor:
  Most of my time 'playing' was spent manually typing the code seen in the notepad into the code editor. This is infuriating. Is this a typing game? I was disappointed at the lack of tab support. I spent lots of time typing spaces. I got confused when doing pg3.cpp. I mainly delve into Java, and was surprised that << is an insertion operator. I thought it's related to bit manipulation. When I tried compiling pg3.cpp, I got a strange message in addition to the error messages expected: "Error 2: File Not Found code.obj" The error message box can grow to be too big, hiding the code editor.
  I liked the little name badge shown on the desk. I thought there would be more days to play, so I was disappointed when the game ended and I got only 57%.”

7 Επίλογος

7.1 Σύνοψη και συμπεράσματα

Στην σημερινή, γρήγορη εποχή η Πληροφορική δείχνει να βρίσκεται πάντα ένα βήμα μπροστά, προσφέροντας συνεχώς νέες λύσεις και βελτιώνοντας τις ήδη υπάρχουσες. Ωστόσο, δεν είναι μόνο ο τομέας της Πληροφορικής που βελτιώνεται, πολλές φορές θεαματικά, σε όλους τους τομείς. Στη σημερινή εποχή της Πληροφορικής, υπάρχουν πλέον πολλές απαιτήσεις από αυτήν.

Ο άνθρωπος “διαθέτει” πολύ λιγότερο χρόνο απ’ ότι στο παρελθόν και πρέπει να κάνει πολλά περισσότερα σε λιγότερο χρόνο. Πλέον, οι καινοτόμες ιδέες γεννιούνται η μια πίσω από την άλλη, όχι γιατί εξελιχθήκαμε σε εξυπνότερα όντα αλλά οι ανάγκες για ταχύτητα και χρήση Διαδικτύου αυξάνουν συνεχώς. Πλέον ο άνθρωπος προτιμάει να τα κάνει όλα από το γραφείο του σπιτιού του ή το κινητό του, με το πάτημα ενός κουμπιού.

Στον τομέα της ηλεκτρονικής εκπαίδευσης, υπάρχουν πλέον τόσα πολλά να ασχοληθεί κάποιος και να εκπαιδευτεί επάνω σε αυτά, όσες είναι περίπου και οι *ανάγκες* τις οποίες καλύπτουν. Το μόνο που έχει να κάνει κάποιος είναι απλώς να το επιλέξει και να το κάνει. Μια από τις ανάγκες αυτές είναι και η εκπαίδευση με διασκεδαστικό τρόπο, τα λεγόμενα *παιχνίδια σοβαρού σκοπού.* Με άλλα λόγια γιατί να εκπαιδευτεί κάποιος σε κάτι εάν δεν το διασκεδάζει παράλληλα.

Σκοπός της διπλωματικής εργασίας ήταν η ανάπτυξη μιας εφαρμογής σοβαρού σκοπού, η οποία θα εκπαιδεύει τους χρήστες στη γλώσσα προγραμματισμού C++. Το παιχνίδι σχεδιάστηκε κατάλληλα ώστε να έχει τη μορφή ενός προσομοιωτή εργασίας & προγραμματισμού. Στοχεύει σε χρήστες με βασικές γνώσεις προγραμματισμού, με κύριο σκοπό να τους εισάγει στη γλώσσα προγραμματισμού `C++`. Ωστόσο, επιλέχθηκε μια άλλη προσέγγιση, εισάγοντας τη διάσταση του *ρεαλισμού* & *υπευθυνότητας* στα χαρακτηριστικά μιας εκπαιδευτικής εφαρμογής σοβαρού σκοπού.

Το πρώτο συμπέρασμα από την εργασία είναι ότι ήταν αρκετά δύσκολη και χρονοβόρα στην ολοκλήρωση της. Η ανάπτυξη της εφαρμογής σοβαρού σκοπού, η οποία πραγματοποιήθηκε παράλληλα με την ανάπτυξη και χρήση της προσαρμοσμένης μηχανής παιχνιδιών στην οποία στηρίχθηκε, αποτέλεσε από μόνο του ένα εξαιρετικά δύσκολο & χρονοβόρο εγχείρημα.

Το δεύτερο συμπέρασμα αφορά στην εκπαιδευτική προσέγγιση. Επιλέχθηκε ένας τρόπος κατά τον οποίο η διαδικασία εκπαίδευσης δεν θα ήταν ιδιαίτερα διασκεδαστική, ωστόσο η παρουσίαση, ο ρεαλισμός & η υπευθυνότητα που θα έπρεπε να επιδείξει ο χρήστης κατά την χρήση, θα αναπλήρωναν το κενό του *διασκεδαστικού* παράγοντα σε τέτοιο βαθμό που ίσως να το καθιστούσαν και εθιστικό μετά από κάποιες χρήσεις.

Ένα ακόμη συμπέρασμα είναι ότι ένα παιχνίδι που ακολουθεί τέτοια προσέγγιση, θα πρέπει να είναι τεχνικά άρτιο, ενώ οι λειτουργίες, η απόδοση & ο ρεαλισμός του να βρίσκονται σε υψηλό επίπεδο. Αγνοώντας αρκετά από τα συνήθη *διασκεδαστικά* χαρακτηριστικά άλλων εφαρμογών σοβαρού σκοπού, στο παιχνίδι αυτό έπρεπε να αποδοθούν σε υψηλό επίπεδο όλα τα υπόλοιπα χαρακτηριστικά που προαναφέρθηκαν.

Τα συμπεράσματα από την πιλοτική αξιολόγηση της εφαρμογής ήταν εξαιρετικά χρήσιμα, καθώς υπέδειξαν τους τομείς στους οποίους η εφαρμογή τα πήγε περίφημα αλλά και που διαπιστώθηκαν αδυναμίες. Με βάση το πλαίσιο αξιολόγησης MEEGA, το παιχνίδι τα πήγε καλά στις περισσότερες διαστάσεις.

Ωστόσο, διαπιστώθηκαν σχεδιαστικές αδυναμίες στις διαστάσεις της *πρόκλησης*, *εμβύθησης* και *μακροπρόθεσμης μάθησης*. Οι αδυναμίες αυτές, μεταξύ άλλων, προκαλούν μονοτονία στο παιχνίδι, έλλειψη συγκέντρωσης στους παίχτες καθώς και αίσθηση ότι δεν υπάρχει αποτέλεσμα από τη χρήση της εφαρμογής. Ωστόσο, αναφέρθηκαν τρόποι αντιμετώπισης των ζητημάτων αυτών στο κεφάλαιο της πιλοτικής αξιολόγησης και θα ληφθούν υπόψη σε μελλοντικές εκδόσεις της εφαρμογής.

7.2 Όρια και περιορισμοί της έρευνας

Στην εποχή που βρισκόμαστε, όλες οι εφαρμογές που αναπτύσσονται θα πρέπει να μπορούν να εκτελεστούν παντού, σε κάθε υπολογιστή, σε κάθε κινητή συσκευή, στην κονσόλα, καθώς και στην τηλεόραση του σαλονιού.

Η εφαρμογή αναπτύχθηκε με τέτοιο τρόπο ώστε να μπορεί να μεταγλωττιστεί στα περισσότερα λειτουργικά συστήματα και συσκευές που υπάρχουν σήμερα στον κόσμο. Ωστόσο, κάποιες λειτουργίες περιόρισαν αρκετά την εφαρμογή και στον βαθμό που μπορεί να το επιτύχει αυτό.

Για παράδειγμα, η δυνατότητα δυναμικής μεταγλώττισης & εκτέλεσης ενός πηγαίου κώδικα που έχει συντάξει ο χρήστης, την ώρα που εκτελείται η εφαρμογή σοβαρού σκοπού δεν είναι κάτι που θα μπορούσε να γίνει εύκολα σε συστήματα που δεν έχουν ευμέγεθες πληκτρολόγιο ή μεταγλωττιστή σε C++ σε μορφή βιβλιοθήκης.

7.3 Μελλοντικές Επεκτάσεις

Η εφαρμογή σοβαρού σκοπού η οποία αναπτύχθηκε, απλώς άνοιξε τον δρόμο για νέες ιδέες & βελτιώσεις. Μερικές από αυτές παρουσιάζονται παρακάτω:

- ✔ **Δυνατότητα μεταγλώττισης κώδικα σε πολλαπλά συστήματα:** Ένα μεγάλο πρόβλημα της εφαρμογής, το οποίο είναι ταυτόχρονα και ένα μεγάλο της πλεονέκτημα, είναι η δυνατότητα να μεταγλωττίζει, σε πραγματικό χρόνο, τον κώδικα που συντάσσει ο χρήστης. Ωστόσο, σε περιορισμένης δυναμικής συστήματα π.χ. Android και iOS κάτι τέτοιο είναι ακόμη πολύ μακριά, καθιστώντας την εφαρμογή κατάλληλη, προς το παρόν, μόνο σε συστήματα τύπου *Windows, Linux* και *Mac OSX*.

- ✔ **Εισαγωγή περισσότερων ρεαλιστικών διαδικασιών:** Σε όλο το κείμενο της εργασίας, η λέξη *ρεαλισμός* αναφέρεται αρκετές φορές και έγινε μεγάλη προσπάθεια ώστε η εφαρμογή να περιέχει αρκετά ρεαλιστικά στοιχεία. Ωστόσο, αποδείχθηκε ότι απλώς και μόνο η σωστή & λειτουργική ανάπτυξη του κυρίως παιχνιδιού δεν άφησαν χρονικά περιθώρια για κάτι παραπάνω. Στο μέλλον υπάρχουν σχέδια για διαδικασίες οι οποίες θα έχουν στόχο να *αποπροσανατολίζουν* τον χρήστη σε τυχαίες στιγμές ώστε να βρίσκεται σε συνεχή εγρήγορση. Η προσθήκη επίσης αρκετών διαδικασιών σε μορφή μινι-παιχνιδιών (mini-games) βρίσκονται στα σχέδια αυτά.

- ✔ **Προσθήκη γλωσσών προγραμματισμού:** Η εκπαίδευση στην γλώσσα προγραμματισμού C++ έγινε επιλεκτικά λόγω γνώσης & εμπειρίας. Ωστόσο, η εφαρμογή δεν απαιτεί κάτι τέτοιο. Η χρήση συγκεκριμένου

μεταγλωττιστή, καθώς και το εκπαιδευτικό περιεχόμενο είναι αυτά που ορίζουν την μορφή της εφαρμογής. Εάν για παράδειγμα η εφαρμογή εκπαίδευε στην γλώσσα Java, θα αρκούσε απλώς η αλλαγή του μεταγλωττιστή & το αρχείο περιεχομένου.

- ✔ **Δυνατότητα εξέλιξης στην εταιρεία:** Αφορούσε μια απαίτηση στις προδιαγραφές της εφαρμογής, η οποία ωστόσο δεν κατέστη δυνατόν να υλοποιηθεί, κυρίως χρονικά. Ωστόσο, στο μέλλον υπάρχει σοβαρή σκέψη να αναπτυχθεί μια επέκταση κατά την οποία ο εικονικός εργαζόμενος θα μπορεί να εξελίσσεται σε διάφορες θέσεις ή να αλλάζει και αντικείμενο π.χ. άλλη γλώσσα προγραμματισμού ή ακόμη να δέχεται προσφορές και από άλλες εταιρείες.

- ✔ **Διαδικτυακό παιχνίδι:** Μια εφαρμογή σοβαρού σκοπού θα μπορούσε να είναι αρκετά πιο διασκεδαστική εάν τα αποτελέσματα δεν τα έβλεπε μόνο ο ίδιος ο χρήστης αλλά παράλληλα και άλλοι χρήστες, οδηγώντας σε πολλαπλά οφέλη.

- ✔ **Διαφορετικά σενάρια:** Παραπάνω αναφέραμε για προσθήκη γλωσσών προγραμματισμού. Ωστόσο, το ύφος του παιχνιδιού δεν έχει να κάνει μόνο με προγραμματισμό, απλώς αποτελεί μια καλή εφαρμογή του. Θα μπορούσε να γίνει προσθήκη *σεναρίων* όπου ο χρήστης καλείται να κάτσει μπροστά σε έναν υπολογιστή και να ολοκληρώσει απλά κάποιες εργασίες που απαιτούν πληκτρολόγηση. Κάποιες ιδέες αφορούν συντακτικό έλεγχο σε εκθέσεις, διόρθωση αιτήσεων κλπ.

Βιβλιογραφία

Csikszentmihalyi, M. (1975). *Beyond Boredom and Anxiety*. San Francisco, USA: Jossey-Bass Publishers, pp.1-13, 34-54.

Djaouti, D., Alvarez, J., Jessel, J.P., & Rampnoux, O. (2011). *Origins of Serious Games*. Ανακτήθηκε από: http://www.ludoscience.com

Hayes, B.M., & Aspray, W. (Eds.). (2010). *Health Informatics: A Patient-centered Approach to Diabetes.* Cambridge, Massachusetts: The MIT Press, p.135.

Kirch-Prinz, U., & Prinz, P. (2002). *A Complete Guide to Programming in C++.* Sudburry, Massachusetts: Jones and Bartlett Publishers, Inc.

Song, M., & Zhang, S. (2008). *EFM: A Model for Educational Game Design.* College of Education, Zhejiang Normal University, China.

Savi, R., Wangenheim, G. C., & Borgatto, F. A. (2011). *MEEGA: A Model for the Evaluation of Educational Games for Teaching Software Engineering.* Conference: 25th Brazilian Symposium on Software Engineering (SBES), Sao Paulo, Brazil.

Petri, G., Wangenheim., G. C., & Borgatto F. A. (2016). *MEEGA+: An Evolution of a Model for the Evaluation of Educational Games.* Brazilian Institute for Digital Convergence, Federal University of Santa Catarina, Brazil.

Petri, G., Wangenheim., G. C., & Borgatto F. A. (2018). *MEEGA+: A Method for the Evaluation of Educational Games for Computing Education.* Brazilian Institute for Digital Convergence, Federal University of Santa Catarina, Brazil.

The Texas A&M University System. (2017). *ARCS Model of Motivation*. Ανακτήθηκε από: http://www.tamus.edu

Basili, R. V., Caldiera, G., & Rombach D. H. (1994). *The Goal Question Metric Approach*. Institude for Advanced Computer Studies, Department of Computer Science, University of Maryland. Ανακτήθηκε από: https://www.cs.umd.edu

Zyda, M. (2005). *From Visual Simulation to Virtual Reality to Games.* pp.25-32, IEEE Computer Society, Information Sciences Institute, California.

Abt, C. (1970). *Serious Games*. University Press of America (επανέκδοση 2002)

Laporte, L., Zaman, B., & Grooff D. D. (2013). *Exploring the value of genres in serious games.* iMinds Research Institute, Belgium.

Gloria, D. A., Bellotti, F., Berta, R., & Lavagnino E. (2014). *Serious Games for Education and Training.* University of Genoa, International Journal of Serious Games. Ανακτήθηκε από: https://journal.seriousgamessociety.org

The Flow Centre. (2016, Ιούνιος). *9 Dimensions to Flow*. Ανακτήθηκε από: https://theflowcentre.com/9-dimensions-to-flow

Rawitsch, D. (2017). *Classic Game Postmortem: Oregon Trail*. GDC, Game Developers Conference, Moscone Center, San Francisco, USA.

Stack Overflow – Where Developers Learn, Learn, & Build Careers (2019). *Stack Overflow Developer Survey Results.* Ανακτήθηκε από: https://insights.stackoverflow.com/survey/2019

Wikipedia, the free encyclopedia (2019). *America's Army*. Ανακτήθηκε από: https://en.wikipedia.org/wiki/America%27s_Army

Official Colobot: Gold Edition Website – International Colobot community (2019). *Learn More.* Ανακτήθηκε από: https://colobot.info/learn-more