ΠΑΝΕΠΙΣΤΗΜΙΟ ΜΑΚΕΔΟΝΙΑΣ
ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ
ΤΜΗΜΑΤΟΣ ΕΦΑΡΜΟΣΜΕΝΗΣ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΣΧΕΔΙΑΣΗ, ΑΝΑΠΤΥΞΗ ΚΑΙ ΑΞΙΟΛΟΓΗΣΗ ΠΑΙΧΝΙΔΙΟΥ ΣΟΒΑΡΟΥ ΣΚΟΠΟΥ ΓΙΑ ΤΗΝ ΕΚΜΑΘΗΣΗ ΤΗΣ ΓΛΩΣΣΑΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ ΡΥΤΗΟΝ

Διπλωματική Εργασία

του

Τσούντα Γεώργιου

Θεσσαλονίκη, Φεβρουάριος 2021

ΣΧΕΔΙΑΣΗ, ΑΝΑΠΤΥΞΗ ΚΑΙ ΑΞΙΟΛΟΓΗΣΗ ΠΑΙΧΝΙΔΙΟΥ ΣΟΒΑΡΟΥ ΣΚΟΠΟΥ ΓΙΑ ΤΗΝ ΕΚΜΑΘΗΣΗ ΤΗΣ ΓΛΩΣΣΑΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ ΡΥΤΗΟΝ

Τσούντας Γεώργιος

Πτυχίο Οργάνωσης και Διοίκησης Επιχειρήσεων, Πανεπιστήμιο Μακεδονίας, 2019

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 26/02/2021

Ξυνόγαλος Στυλιανός	Σατρατζέμη Μαρία	Βεργίδης Κωνσταντίνος
.....	.....	.....

Τσούντας Γεώργιος

.....

Περίληψη

Στην παρούσα διπλωματική εργασία αναπτύχθηκε ένα παιχνίδι σοβαρού σκοπού που στοχεύει στην εκμάθηση της γλώσσας προγραμματισμού Python. Το συγκεκριμένο παιχνίδι δεν απαιτεί κάποιο προγραμματιστικό υπόβαθρο από τους παίκτες καθώς αναλαμβάνει να τους διδάξει εισαγωγικές αλλά και πιο προχωρημένες λειτουργίες της γλώσσας προγραμματισμού Python. Πριν την υλοποίηση, αναλύονται χαρακτηριστικά και θέματα παιχνιδιών σοβαρού σκοπού με εκπαιδευτικό περιεχόμενο σε σχέση με την διδασκαλία και την εκμάθηση της γλώσσας προγραμματισμού Python. Το παιχνίδι σχεδιάστηκε ώστε να συνδυάζει την δημιουργία κώδικα Python που θα παράγει οπτικό αποτέλεσμα. Με βάση αυτήν την ανάλυση και σχεδίαση υλοποιείται το παιχνίδι, το οποίο αναπτύχθηκε στη μηχανή παιχνιδιών Unreal Engine 4. Τέλος, αξιολογείται το παιχνίδι βάσει ερωτηματολογίου αξιολόγησης παιχνιδιών σοβαρού σκοπού από φοιτητές που εθελοντικά συμμετείχαν και έπαιξαν ώστε να εξαχθούν συμπεράσματα για την τελική μορφή του παιχνιδιού και το ποσοστό επίτευξης των στόχων που ορίστηκαν πριν την δημιουργία του.

Λέξεις Κλειδιά: παιχνίδια σοβαρού σκοπού, Python, εκμάθηση Python

Abstract

In this thesis, a serious game has been created that aims in teaching the Python programming language. This game does not require any prior programming knowledge from the players since it is responsible for the teaching from a beginner to a more advanced level of the Python programming language. Before creating the game, characteristics and subjects of serious games with educational content in relation to teaching the Python programming language are analyzed. The game was designed to combine the creation of Python code that would generate visual results. Based on that analysis and design the final game is created and it has been developed using the game engine Unreal Engine 4. Finally, the game is evaluated through a specially designed questionnaire by undergraduate and postgraduate students, who participated voluntarily and played the game in order to draw conclusions for the final form of the game and the degree of achieving the goals set before its creation.

Keywords: serious games, Python, learn Python

Ευχαριστίες

Ευχαριστώ τον επιβλέπων καθηγητή κ. Ξυνόγαλο Στέλιο για τη βοήθεια και τις συμβουλές που μου έδωσε αλλά και για την υπομονή και το ενδιαφέρον του προς την ολοκλήρωση της διπλωματικής εργασίας μου.

Περιεχόμενα

Κατάλογος Εικόνων

Κατάλογος Πινάκων

Κώδικες

Κατάλογος Blueprint

1 Εισαγωγή

1.1 Πρόβλημα – Σημαντικότητα του θέματος

Πολλές φορές αποδεικνύεται δύσκολο να παρουσιαστεί μία σύνθετη έννοια ή ένα περίπλοκο θέμα σε άλλους ανθρώπους ώστε αυτοί να είναι σε θέση να το αντιληφθούν με τρόπο που θα τους επιτρέψει να το εμπεδώσουν. Ιδιαίτερα δύσκολο γίνεται όταν τα άτομα αυτά δεν έχουν καμία προηγούμενη εμπειρία ή γνώση για το συγκεκριμένο θέμα. Ένα τέτοιο παράδειγμα αποτελούν οι μαθητές που πολλές φορές δεν έχουν καμία προηγούμενη γνώση για το αντικείμενο που διδάσκονται. Σε τέτοιες περιπτώσεις, έχουν χρησιμοποιηθεί τα ηλεκτρονικά παιχνίδια και έχουν παρατηρηθεί σημαντικά αποτελέσματα βελτίωσης αποκτηθείσας γνώσης (Sousa & Costa, 2018). Τέτοιου τύπου παιχνίδια ονομάζονται παιχνίδια σοβαρού σκοπού.

Τα παιχνίδια σοβαρού σκοπού σε αρκετές περιπτώσεις ορίζονται απλά ως παιχνίδια που δεν έχουν ως μοναδικό σκοπό την ψυχαγωγία και τη διασκέδαση (Michael & Chen, 2006). Ένας άλλος ορισμός για τα παιχνίδια σοβαρού σκοπού (serious games) επισημαίνει ότι πρόκειται για παιχνίδια που συνδυάζουν έναν μη ψυχαγωγικό σκοπό (serious) με μια δομή παιχνιδιού (game) (Djaouti, Alvarez & Jessel, 2011).

Οι μαθητές στον προγραμματισμό πολλές φορές συναντούν δυσκολίες κατά την εκμάθηση ιδιαίτερα σε σχέση με αφηρημένες έννοιες (Gomes & Mendes, 2007). Έτσι, για την διδασκαλία προγραμματισμού, στα πλαίσια ενός παιχνιδιού και χρησιμοποιώντας μια δραστηριότητα στο παιχνίδι που εμφανίζει ομοιότητες με τον μηχανισμό πίσω από μια συγκεκριμένη έννοια προγραμματισμού ως μέσο παρουσίασής της, μπορούμε να βελτιώσουμε την κατανόησή της από τους μαθητευόμενους (Zapušek & Rugelj, 2013). Με αυτή τη λογική και στα πλαίσια ολοκλήρωσης της διπλωματικής εργασίας, θα γίνει προσπάθεια δημιουργίας ενός παιχνιδιού που συμπεριλαμβάνει εκτός από την ψυχαγωγία και την εκμάθηση της γλώσσας προγραμματισμού Python.

1.2 Σκοπός – Στόχοι

Σκοπός της διπλωματικής εργασίας είναι η σχεδίαση και ανάπτυξη ενός παιχνιδιού σοβαρού σκοπού που θα διδάσκει την γλώσσα προγραμματισμού Python. Το παιχνίδι θα ξεκινάει εισάγοντας τους παίκτες στο συντακτικό, τη δομή και τη φιλοσοφία της Python και θα επεκτείνεται σε πιο προχωρημένες λειτουργίες και τρόπο χρήσης της γλώσσας.

Στόχος είναι ο παίκτης στο τέλος του παιχνιδιού να είναι σε θέση να αντιλαμβάνεται και να μπορεί να χρησιμοποιήσει τα περισσότερα βασικά στοιχεία της γλώσσας προγραμματισμού Python. Ακόμη, επιπλέον στόχος είναι το παιχνίδι να προκαλέσει το ενδιαφέρον του παίκτη για την γλώσσα, ενώ επίσης παροτρύνεται η εξερεύνηση του τεράστιου αριθμού πακέτων της και των δυνατοτήτων που προσφέρουν, των οποίων τη λειτουργία θα είναι σε θέση να κατανοήσει και να χρησιμοποιήσει στο μέλλον.

1.3 Διάρθρωση της μελέτης

Στο Κεφάλαιο 2, γίνεται βιβλιογραφική ανασκόπηση σε θέματα παιχνιδιών σοβαρού σκοπού εκπαιδευτικού περιεχομένου αλλά και έρευνα προς εύρεση και ανάλυση παιχνιδιών σοβαρού σκοπού σχετικών με το θέμα της εργασίας καθώς και εξαγωγή τελικών συμπερασμάτων για την σχεδίαση του παιχνιδιού.

Στο Κεφάλαιο 3, παρουσιάζεται η μεθοδολογία που χρησιμοποιήθηκε για τη σχεδίαση του παιχνιδιού και η λογική που ακολουθείται κατά την υλοποίησή του. Επίσης, αναλύονται ζητήματα πριν τη δημιουργία του παιχνιδιού και αποφάσεις που λήφθηκαν.

Στο Κεφάλαιο 4, αναλύεται το παιχνίδι που αναπτύχθηκε και παρουσιάζεται ο τρόπος ένταξης θεμάτων που εξετάστηκαν στα προηγούμενα κεφάλαια, εντός του παιχνιδιού.

Στο Κεφάλαιο 5, παρουσιάζεται το ολοκληρωμένο παιχνίδι και τα επιμέρους περιεχόμενα σε σχέση με το αντικείμενο που προσπαθεί να διδάξει.

Στο Κεφάλαιο 6, γίνεται παρουσίαση της διαδικασίας και των συστημάτων λειτουργίας που αναπτύχθηκαν για την τελική υλοποίηση του παιχνιδιού.

Στο Κεφάλαιο 7, γίνεται αξιολόγηση του παιχνιδιού και παρουσιάζονται τα αποτελέσματά της.

Στο Κεφάλαιο 8, παρουσιάζονται τα συμπεράσματα που προκύπτουν με βάση την εργασία και αναλύονται οι πιθανές μελλοντικές επεκτάσεις της.

2 Βιβλιογραφική Επισκόπηση – Θεωρητικό Υπόβαθρο

2.1 Παιχνίδια σοβαρού σκοπού εκπαιδευτικού περιεχομένου

Ένας από τους σκοπούς που μπορεί να περιλαμβάνει ένα παιχνίδι εκτός από την ψυχαγωγία είναι η εκπαίδευση σε κάποιο θέμα. Η ιστοσελίδα (http://serious.gameclassification.com) αποτελεί μια βάση δεδομένων κατηγοριοποίησης παιχνιδιών σοβαρού σκοπού. Σε αναζήτηση παιχνιδιών στα 3399 παιχνίδια που υπάρχουν καταχωρημένα, βρέθηκαν 1893 που περιλαμβάνουν εκπαιδευτικούς σκοπούς (Serious Game Classification : The online classification of Serious Games, n.d.).

Τα παιχνίδια σοβαρού σκοπού με εκπαιδευτικό περιεχόμενο μπορούν να προσφέρουν πλεονεκτήματα ιδιαίτερα όταν συμπεριλαμβάνονται στα πλαίσια εκπαίδευσης σε αίθουσες διδασκαλίας ως εκπαιδευτικό εργαλείο (Michael & Chen, 2006). Όμως, πολλές φορές αποδεικνύεται δύσκολη η ενσωμάτωσή τους σε εκπαιδευτικά προγράμματα. Οι Fernández-Manjón, Moreno-Ger, Martinez-Ortiz και Freire (2015) προτείνουν, μεταξύ άλλων, τρόπους βελτίωσης ανάπτυξης και ένταξης παιχνιδιών σοβαρού σκοπού στην εκπαίδευση:

- Δημιουργία οδηγών σχετικά με το παιχνίδι και τα περιεχόμενά του
- Μείωση πολυπλοκότητας και κόστους ανάπτυξης παιχνιδιών σοβαρού σκοπού εκπαιδευτικού περιεχομένου με χρήση σύγχρονων μεθόδων
- Μείωση τεχνολογικών απαιτήσεων χρήσης των παιχνιδιών

Δεν υπάρχουν συγκεκριμένα στοιχεία και χαρακτηριστικά παιχνιδιών σοβαρού σκοπού που είναι απαραίτητα για την εκπαίδευση αλλά όταν πληροφορίες και περιεχόμενο συνδυάζονται με τα κατάλληλα χαρακτηριστικά παιχνιδιών, τότε οι παίκτες έχουν κίνητρο να ασχοληθούν με αυτά (Wilson et al., 2008). Όμως, κάθε παιχνίδι σοβαρού σκοπού με εκπαιδευτικό περιεχόμενο μπορεί να συνδυάσει διαφορετικά χαρακτηριστικά εκπαίδευσης με στοιχεία παιχνιδιού ανάλογα με τους στόχους και τα επιθυμητά αποτελέσματα αλλά και τον τρόπο χρήσης του ώστε να είναι πιο αποτελεσματικό (Lameras et al., 2017).

2.2 Παιχνίδια σοβαρού σκοπού για την Python

Πραγματοποιήθηκε έρευνα για την εύρεση παιχνιδιών σοβαρού σκοπού για τη γλώσσα προγραμματισμού Python ή παιχνιδιών που υποστηρίζουν Python ανάμεσα σε άλλες γλώσσες. Στη συνέχεια παρουσιάζονται και αναλύονται τα παιχνίδια αυτά.

2.2.1 Code Combat

https://codecombat.com/

Το Code Combat είναι ένα παιχνίδι σοβαρού σκοπού που προσπαθεί να διδάξει προγραμματισμό στον παίκτη. Το παιχνίδι διαδραματίζεται σε φανταστικούς κόσμους και οι παίκτες δημιουργούν κώδικα ώστε να κατευθύνουν τον κεντρικό χαρακτήρα. Τα μαθήματά του διατίθενται για τις γλώσσες προγραμματισμού Python και JavaScript (CodeCombat - Coding games to learn Python and JavaScript, n.d.).

Εικόνα 1: Επίπεδο στο CodeCombat

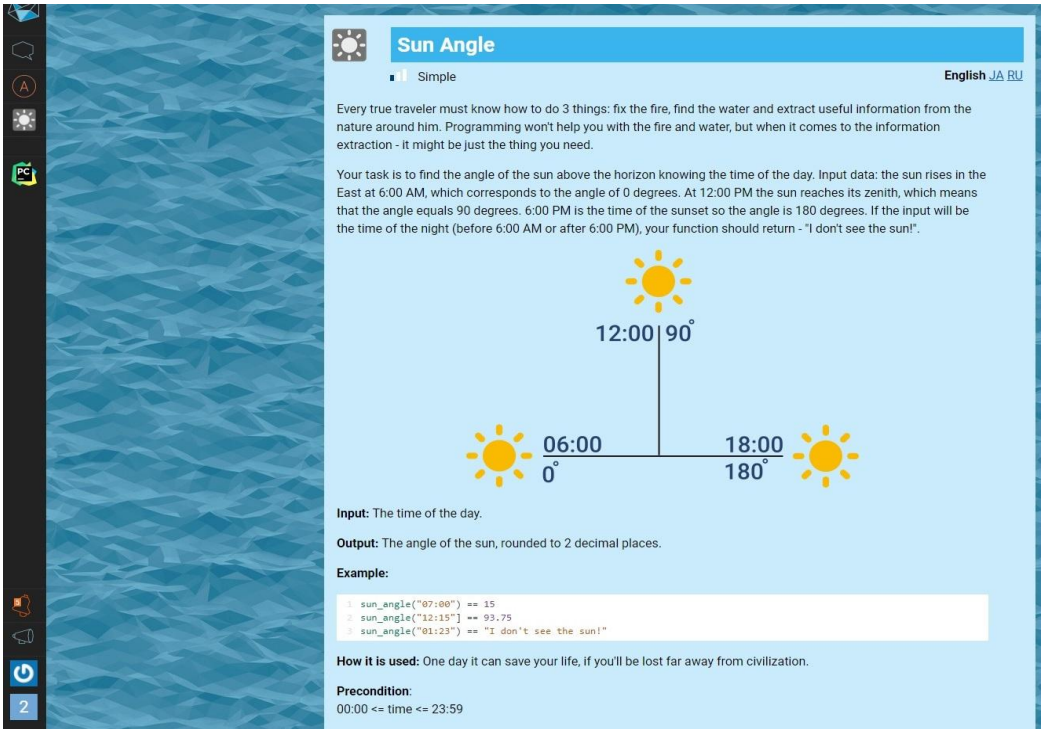
Το CodeCombat, παρέχει την επιλογή ενσωμάτωσής του στο πρόγραμμα διδασκαλίας σχολείων και στους εκπαιδευτικούς την δυνατότητα να δημιουργήσουν τάξεις στις οποίες προσκαλούν τους μαθητές τους. Έτσι, μπορούν να παρακολουθούν την πορεία των μαθητών και να εντοπίζουν τα λάθη τους. Επίσης, με χρήση διαφορετικών επιπέδων και λογικής στις δοκιμασίες που παρουσιάζονται, πιο εξοικειωμένοι στον προγραμματισμό μαθητές μπορούν να μεταβούν σε επίπεδα εξάσκησης των ικανοτήτων τους καθώς οι μαθητές που δυσκολεύονται προσπαθούν να εξοικειωθούν με τον προγραμματισμό (CodeCombat - Coding games to learn Python and JavaScript, n.d.).

Κατά τη διάρκεια του παιχνιδιού, για τους παίκτες και τους μαθητές που δυσκολεύονται να κατανοήσουν τις έννοιες και τα θέματα που παρουσιάζει το παιχνίδι, υπάρχουν βοηθήματα με την μορφή “Hints” όπου παρέχεται επιπλέον βοήθεια και επεξήγηση του θέματος στον παίκτη ώστε αυτός να καταφέρει να ολοκληρώσει την δοκιμασία. Επίσης, το παιχνίδι είναι εύκολα προσβάσιμο σε όποιον παίκτη ενδιαφέρεται και δεν απαιτείται κάποια εγκατάσταση λογισμικού ή κάποιο ιδιαίτερο επίπεδο συστήματος. Το μόνο που είναι απαραίτητο είναι ένας ηλεκτρονικός υπολογιστής με κάποιο πρόγραμμα περιήγησης ιστού και πρόσβαση στο διαδίκτυο (CodeCombat - Coding games to learn Python and JavaScript, n.d.).

2.2.2 CheckiO

https://checkio.org/

Το CheckiO είναι μια ιστοσελίδα με εκπαιδευτικό περιεχόμενο που παρουσιάζεται με τη μορφή παιχνιδιού και στοχεύει οι παίκτες να μάθουν να γράφουν καλύτερο κώδικα. Οι παίκτες με χρήση Python ή JavaScript προσπαθούν να επιλύσουν δοκιμασίες σε πλήθος θεμάτων και τύπους προβλημάτων. Το παιχνίδι προϋποθέτει κάποιες προγραμματιστικές γνώσεις από τον παίκτη ξεκινώντας από κάποιες ευκολότερες προγραμματιστικές δοκιμασίες. Με τη χρήση της διαδικτυακής πλατφόρμας (forum) επικοινωνίας και των σχολίων σε κάθε δοκιμασία οι χρήστες μπορούν να μοιραστούν τις λύσεις τους και να δουν άλλους τρόπους επίλυσης.



Εικόνα 2: Δοκιμασία στο CheckiO

Το CheckiO, έχει τη δυνατότητα ενσωμάτωσής του στα πλαίσια τάξης ως εκπαιδευτικό εργαλείο. Παρέχεται πλατφόρμα χρήσης του για εκπαιδευτικούς με εργαλεία ελέγχου προόδου μαθητών και εύρεσης σημείων δυσκολίας κατανόησης και επίλυσης των προβλημάτων (Teaching with CheckiO ClassRooms, n.d.).

2.2.3 CodeinGame

https://www.codingame.com

Το CodeinGame είναι μια διαδικτυακή πλατφόρμα που διαθέτει παιχνίδια και μαθήματα για προγραμματιστές. Τα περισσότερα παιχνίδια που διαθέτει έχουν τη μορφή γρίφων και κατατάσσονται στην κατηγορία TBS (turn-based strategy) καθώς ο παίκτης σε κάθε γύρο πρέπει να εισάγει κώδικα που θα παράγει τις κατάλληλες στρατηγικές βάσει παραμέτρων που δίνονται. Τα παιχνίδια που διατίθενται ποικίλουν σε θεματολογία και δυσκολία αλλά μπορούν να λυθούν σε οποιαδήποτε από τις υποστηριζόμενες γλώσσες (Coding Games and Programming Challenges to Code Better, n.d.).

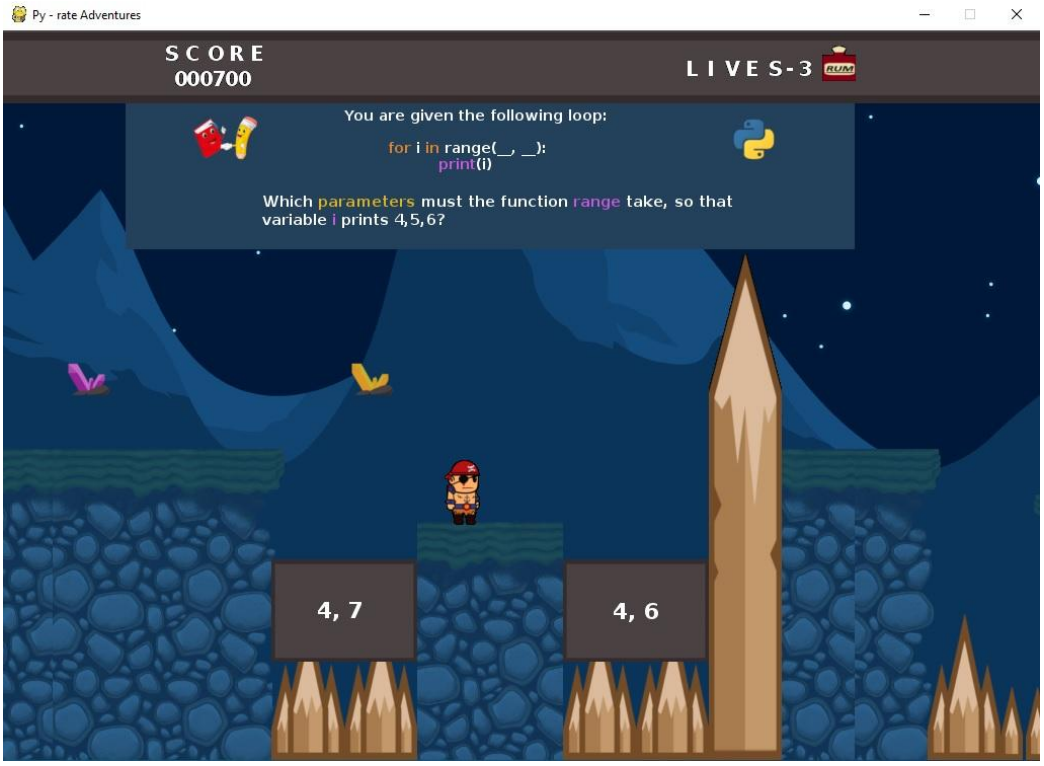
Στο CodeinGame υποστηρίζονται: Bash, C, C#, C++, Clojure, D, Dart, F#, Go, Groovy, Haskell, Java, JavaScript, Kotlin, Lua, Objective-C, OCaml, Pascal, Perl, PHP, Python, Ruby, Rust, Scala, Swift, TypeScript, VB.NET. Ο χρήστης επιλέγει αρχικά την

γλώσσα που θέλει και όλες οι δοκιμασίες στα παιχνίδια είναι σε αυτή τη γλώσσα, αλλά μπορεί οποιαδήποτε στιγμή να αλλάξει γλώσσα και να συνεχίσει παίζοντας στην νέα γλώσσα που επέλεξε (Coding Games and Programming Challenges to Code Better, n.d.).

2.2.4 Py-rate Adventures

Το παιχνίδι Py-rate Adventures (Σιδέρης, 2019) είναι ένα παιχνίδι σοβαρού σκοπού το οποίο σχεδιάστηκε και αναπτύχθηκε κατά τη διάρκεια ολοκλήρωσης διπλωματικής εργασίας. Ο παίκτης είναι στη θέση του κεντρικού χαρακτήρα και τον χειρίζεται για να κινηθεί στα επίπεδα, να αντιμετωπίσει τους εχθρούς και να απαντήσει στις δοκιμασίες που συναντάει.

Κατά την διάρκεια του παιχνιδιού, ο παίκτης δεν δημιουργεί κώδικα και δεν πραγματοποιείται εκτέλεση εντολών του παίκτη σε Python καθώς οι δοκιμασίες έχουν την μορφή πολλαπλής επιλογής. Τέλος, το εκπαιδευτικό περιεχόμενο δεν συσχετίζεται με τον κόσμο και το περιβάλλον ή τον βασικό χαρακτήρα του παιχνιδιού.



Εικόνα 3: Επίπεδο 5 στο Py-rate Adventures

2.3 Four-Dimensional Framework

Το πλαίσιο Four-Dimensional Framework (de Freitas & Oliver, 2006) αναπτύχθηκε για την αξιολόγηση και την εκτίμηση εκπαιδευτικών παιχνιδιών λόγω έλλειψης εργαλείων και πλαισίων που μπορούν να αξιολογήσουν εκπαιδευτικά παιχνίδια (de Freitas & Jarvis, 2006). Το πλαίσιο λαμβάνει υπόψη κατά την αξιολόγηση το εκπαιδευτικό μέρος ενός παιχνιδιού καθώς και το κομμάτι της διασκέδασης σε παιχνίδια σοβαρού σκοπού (de Freitas & Oliver, 2006).

Σύμφωνα με τους de Freitas και Oliver (2006), οι τέσσερις διαστάσεις (four-dimensions) του πλαισίου αναφέρονται σε θέματα γενικού πλαισίου (context), θέματα που αφορούν στο μαθησιακό προφίλ, τις γνώσεις κτλ του μαθητή (learner specification), αναπαράστασης (representation) και παιδαγωγικά (pedagogy).

Οι de Freitas και Oliver (2006) αναλύουν τις τέσσερις διαστάσεις του πλαισίου:

Η διάσταση γενικού πλαισίου (context), σχετίζεται με τον τρόπο και τον τόπο που θα γίνει χρήση του παιχνιδιού και είναι κεντρικός παράγοντας στην αποτελεσματικότητα που μπορεί να έχει. Παράγοντες που μπορεί να περιλαμβάνει είναι η τοποθεσία και το περιβάλλον χρήσης (τάξη, εξωτερική τοποθεσία, αν θα υπάρχει τεχνική υποστήριξη).

Η διάσταση που αναφέρεται στον μαθητή (learner specification) σχετίζεται με τα χαρακτηριστικά του μαθητευόμενου. Εδώ, περιλαμβάνονται θέματα όπως η ηλικία ή η ύπαρξη και το επίπεδο προϋπάρχουσας γνώσης στο αντικείμενο.

Η διάσταση της αναπαράστασης (representation) σχετίζεται με τον τρόπο που παρουσιάζεται ένα παιχνίδι στους παίκτες. Αυτό περιλαμβάνει με ποιο τρόπο τα επίπεδα εμβύθισης (immersion), πιστότητας (fidelity) και διαδραστικότητας (interactivity) ενσωματώνονται στο παιχνίδι.

Η παιδαγωγική διάσταση (pedagogy) περιλαμβάνει την θεωρία και τις τεχνικές διδασκαλίας που χρησιμοποιούνται. Αυτό μπορεί να αναφέρεται στα επιθυμητά μαθησιακά αποτελέσματα καθώς και το είδος των εκπαιδευτικών δραστηριοτήτων.

2.4 Ανάλυση και συγκριτική αξιολόγηση παιχνιδιών σοβαρού σκοπού

Με βάση το πλαίσιο τεσσάρων διαστάσεων (Four-Dimensional Framework) (de Freitas & Oliver, 2006) γίνεται αξιολόγηση και εκτίμηση σε κάθε διάσταση του

πλαισίου, των εκπαιδευτικών παιχνιδιών που παρουσιάστηκαν σχετικά με την εκμάθηση και την εξάσκηση στη γλώσσα προγραμματισμού Python.

2.4.1 Py-rate Adventures

Στη διάσταση γενικού πλαισίου (context), το Py-rate Adventures μπορεί να θεωρηθεί ότι είναι ένα παιχνίδι που δημιουργήθηκε ώστε ο παίκτης να παίζει μόνος του (single player). Είναι ένα παιχνίδι που θα μπορούσε ένας ενδιαφερόμενος παίκτης να παίξει στον προσωπικό του υπολογιστή κατά τον ελεύθερο χρόνο του και όχι σε κάποια τάξη ή αίθουσα διδασκαλίας ως εκπαιδευτικό εργαλείο. Τεχνική υποστήριξη δίνεται στον ενδιαφερόμενο παίκτη από πλευράς αρχείου οδηγιών (instructions) όπου αναφέρονται οι βασικές λειτουργίες του παιχνιδιού και ο τρόπος εκκίνησης. Σε πιθανή περίπτωση αδυναμίας εκτέλεσης ή εκκίνησης του παιχνιδιού ή πιθανού σφάλματος κατά την λειτουργία του δεν παρέχεται επιπλέον υποστήριξη.

Στη διάσταση που αφορά στον μαθητή (learner specification) το παιχνίδι απευθύνεται σε κοινό χωρίς κάποια ιδιαίτερη προγραμματιστική εμπειρία καθώς εισάγει τον παίχτη στον προγραμματισμό με την Python (Sideris and Xinogalos, 2019). Επίσης, με τον απλό τρόπο λειτουργίας του (gameplay) δεν υπάρχει περιορισμός ηλικίας, με την προϋπόθεση ότι οι παίκτες γνωρίζουν βασικές μαθηματικές και λογικές έννοιες.

Στη διάσταση αναπαράστασης (representation), γίνεται χρήση κινούμενων και στατικών εικόνων, τόσο για τον κεντρικό χαρακτήρα και τους εχθρούς όσο και για το περιβάλλον και το εκπαιδευτικό περιεχόμενο. Από πλευράς διαδραστικότητας ο παίκτης με τις βασικές κινήσεις του χαρακτήρα (άλματα, κίνηση) αλληλοεπιδρά με το περιβάλλον και το εκπαιδευτικό περιεχόμενο.

Στην παιδαγωγική διάσταση (pedagogy), έχει επιλεχθεί η παρουσίαση θεωρίας μέσα από εικόνες που συναντάει ο παίκτης κατά την διάρκεια του παιχνιδιού. Δεν υπάρχει περιορισμός χρόνου και ο παίκτης μπορεί να έχει την εικόνα θεωρίας όσο χρειάζεται στην οθόνη του. Μόλις ο παίκτης θεωρεί ότι διάβασε και κατανόησε το περιεχόμενο που παρουσιάζεται, μπορεί να προχωρήσει στις δοκιμασίες και τους γρίφους που ακολουθούν. Οι ερωτήσεις είναι τύπου πολλαπλής επιλογής όμως, υπάρχει πιθανότητα ο παίκτης να διάλεξε μια τυχαία απάντηση και δεν επεξηγείται το αποτέλεσμα της επιλογής του σε αυτόν. Σε κάθε περίπτωση στον παίκτη δεν δίνεται περιθώριο λάθους απάντησης στις ερωτήσεις.

2.4.2 Code Combat

Το Code Combat μπορεί να χρησιμοποιηθεί είτε από κάποιον ενδιαφερόμενο για προσωπική χρήση και βελτίωση ικανοτήτων κατά τον ελεύθερο χρόνο σε προσωπικό υπολογιστή είτε ως εκπαιδευτικό εργαλείο στα πλαίσιο τάξης ή μαθήματος. Εκπαιδευτικοί μπορούν να δημιουργήσουν εικονική τάξη, στην οποία θα συμμετέχουν οι μαθητές και τους παρέχεται δυνατότητα εποπτείας της καθώς και της προόδου των μαθητών (CodeCombat - Coding games to learn Python and JavaScript, n.d.). Επίσης, υπάρχει υποστήριξη τόσο σε θέματα τεχνικής φύσεως όσο και σε θέματα περιεχομένου από την ομάδα ανάπτυξης και συντήρησης του Code Combat.

Δεν υπάρχει κάποια συγκεκριμένη προγραμματιστική εμπειρία που είναι απαραίτητη για να ξεκινήσει ένας παίκτης στο Code Combat. Μέσα από τις δραστηριότητες του παιχνιδιού διδάσκονται όλες οι αναγκαίες προγραμματιστικές έννοιες που χρησιμοποιούνται και αποτελούν τη βάση όπου στηρίζονται τα επόμενα επίπεδα και το εκπαιδευτικό περιεχόμενο που παρέχουν. Το παιχνίδι απευθύνεται σε οποιονδήποτε ενδιαφέρεται να μάθει προγραμματισμό στις υποστηριζόμενες γλώσσες προγραμματισμού ακόμα και άτομα νεαρότερης ηλικίας (CodeCombat - Coding games to learn Python and JavaScript, n.d.). Επίσης, με την υποδομή που υπάρχει, παίκτες μπορεί να είναι και μαθητές ως ομάδα τμήματος ή τάξης.

Η αναπαράσταση και το σενάριο του παιχνιδιού γίνονται σε κόσμους και επίπεδα δύο διαστάσεων, όμως υπάρχει δυνατότητα κίνησης και εξερεύνησης σε οποιαδήποτε από αυτές. Ο παίκτης λαμβάνει συνεχώς ανατροφοδότηση, σε σχέση με τις επιλογές του, από το παιχνίδι.

Η εκπαιδευτική διαδικασία, έχει τον τρόπο εκμάθησης της φιλοσοφίας και του τρόπου σύνταξης των εντολών, στη γλώσσα προγραμματισμού που επιλέχθηκε. Σε κάθε επίπεδο δίνονται στον παίκτη οδηγίες για την σωστή επίλυση του προβλήματος και το παιχνίδι καθιστά εύκολο τον εντοπισμό λαθών ακόμα και χωρίς την επίβλεψη κάποιου διδάσκοντα. Τέλος, η επίλυση των γρίφων και των προβλημάτων, μπορεί να βοηθήσει στη ανάπτυξη αλγοριθμικής σκέψης που είναι πιθανό ο παίκτης να μην έχει κατά την εκκίνηση του παιχνιδιού. Σε οποιοδήποτε επίπεδο του παιχνιδιού ο παίκτης μπορεί να δοκιμάζει συνεχώς απαντήσεις μέχρι να βρει την λύση και μπορεί να συμβουλεύεται τις «βοήθειες» (hints) που υπάρχουν για το συγκεκριμένο επίπεδο.

2.4.3 CodeinGame

Στο CodeinGame, το γενικό πλαίσιο (context) είναι η προσωπική ενασχόληση του κάθε παίκτη. Η πρόσβαση στα παιχνίδια της πλατφόρμας γίνεται με τη δημιουργία και χρήση προσωπικού λογαριασμού και παρόλο που η δομή και λειτουργία των διαθέσιμων παιχνιδιών δίνει τη δυνατότητα χρήσης σε περιβάλλον τάξης ή στα πλαίσια μαθήματος, δεν παρέχεται αυτή η λειτουργικότητα.

Υπάρχει αριθμός διαθέσιμων παιχνιδιών που μπορεί ένας ενδιαφερόμενος να παίξει όμως όλα προϋποθέτουν κάποιες βασικές γνώσεις προγραμματισμού, ακόμα και αυτά που χαρακτηρίζονται από την πλατφόρμα ως «εύκολα». Το προφίλ των πιθανών παικτών είναι άτομα εξοικειωμένα είτε λιγότερο είτε περισσότερο με τον προγραμματισμό σε κάποια γλώσσα και ενδιαφέρονται να βελτιώσουν τις ικανότητές τους σε αυτήν ή να μάθουν μια καινούρια. Αυτό έχει ως αποτέλεσμα να μην υπάρχει κάποιος ηλικιακός περιορισμός αλλά περισσότερο περιορισμοί προϋπαρχόντων γνώσεων.

Στα περισσότερα διαθέσιμα παιχνίδια, η αναπαράσταση του κόσμου και του σεναρίου γίνεται με τη χρήση κάποιων κινούμενων εικόνων σε χώρο δύο διαστάσεων (2D) και η ιστορία μεταφέρεται στον παίκτη με τη χρήση κειμένου. Κατά τη διάρκεια του παιχνιδιού, η εισαγωγή κώδικα και η δημιουργία απαντήσεων έχει ορατό στον παίκτη αποτέλεσμα. Τέλος, ο παίκτης αλληλεπιδρά με το παιχνίδι βάσει του κώδικα που δημιουργεί και εκτελεί, λαμβάνοντας ανατροφοδότηση από αυτό.

Τα παιχνίδια που είναι διαθέσιμα στην πλατφόρμα, παρουσιάζουν το πρόβλημα στον παίκτη και αυτός καλείται να δημιουργήσει κώδικα που θα αποτελέσει την κίνησή του στο παιχνίδι και θα του επιτρέψει να προχωρήσει στο επόμενο επίπεδο. Επίσης, στον παίκτη δίνονται βοήθειες για την επίλυση των προβλημάτων. Μέσα από την παρουσίαση των αποτελεσμάτων του κώδικα του παίκτη, αυτός πρέπει να κατανοήσει τα λάθη του και να προσαρμόσει τις απαντήσεις του ώστε να προχωρήσει στο παιχνίδι. Τέλος, η δυσκολία των επιπέδων των παιχνιδιών ποικίλει και σε κάθε επίπεδο, ο παίκτης, πρέπει να βρει λύση σε διαφορετικού τύπου προβλήματα.

2.4.4 CheckiO

Στη διάσταση γενικού πλαισίου (context), το CheckiO μπορεί να χρησιμοποιηθεί είτε στα πλαίσια ενδιαφέροντος του κάθε παίκτη είτε ως εκπαιδευτικό εργαλείο στα

πλαίσια τάξεως. Τέλος, δεν μπορεί να θεωρηθεί ότι παρέχεται τεχνική υποστήριξη στους παίκτες στην περίπτωση που συναντήσουν τεχνικά προβλήματα.

Σε σχέση με το προφίλ των μαθητών, παρόλο που το παιχνίδι ξεκινάει από εισαγωγικό επίπεδο, προϋποθέτει προγραμματιστική εμπειρία από τον παίκτη. Επίσης, δεν μπορεί να θεωρηθεί ότι το προφίλ των παικτών θα ήταν άτομα μικρότερης ηλικίας και οι παίκτες που προσελκύονται από το παιχνίδι, προτιμούν την εκμάθηση προγραμματισμού μέσα από ασκήσεις και προβλήματα σε πραγματικό προγραμματιστικό περιβάλλον και όχι τόσο κάποιο περιβάλλον παιχνιδιού με την κλασσική έννοια.

Η αναπαράσταση του παιχνιδιού γίνεται σε χάρτη δύο διαστάσεων με ελάχιστες κινούμενες εικόνες. Η εξέλιξη της ιστορίας με την πρόοδο του παιχνιδιού γίνεται με τη μορφή κειμένων στα επίπεδα που υπάρχουν ενώ υπάρχουν και επίπεδα χωρίς σχετική ιστορία. Στον παίκτη δεν δίνεται κάποιο ιδιαίτερο οπτικό ερέθισμα σε σχέση με τις απαντήσεις του στο γραφικό περιβάλλον του παιχνιδιού.

Η εκπαιδευτική διαδικασία που ακολουθείται γίνεται με τη μορφή προβλημάτων προς επίλυση. Στον παίκτη δίνονται κάθε φορά οδηγίες σχετικά με το τι πρέπει να κάνει προγραμματιστικά για να τα επιλύσει. Ο παίκτης καθώς προγραμματίζει την λύση είναι ελεύθερος να δοκιμάσει όσες φορές χρειαστεί τον κώδικα που έγραψε ώστε να κατανοήσει τα λάθη του και μέσα από τον έλεγχο ολοκλήρωσης του προβλήματος να εντοπίσει τις αστοχίες του και να τις διορθώσει. Κατά τον έλεγχο των απαντήσεων, δίνεται σαφής ανατροφοδότηση σχετικά με τα αναμενόμενα αποτελέσματα αλλά και με τα αποτελέσματα που δημιουργεί ο κώδικας του παίκτη. Τέλος, δεν παρέχεται κάποιου είδους βοήθεια προς τον παίκτη σχετικά με την σωστή απάντηση.

2.4.5 Συγκριτική αξιολόγηση παιχνιδιών

Τα παιχνίδια που αναλύθηκαν και αξιολογήθηκαν με βάση το πλαίσιο τεσσάρων διαστάσεων σε κάθε διάστασή του, έχουν ως στόχο τη διδασκαλία της γλώσσας προγραμματισμού Python, είτε ως κεντρικό εκπαιδευτικό περιεχόμενο είτε ως επιλογή ανάμεσα σε άλλες γλώσσες.

Τα αποτελέσματα της ανάλυσης συνοψίζονται στον Πίνακα 1.

Πίνακας 1: Σύγκριση παιχνιδιών βάσει των διαστάσεων του Four Dimensional Framework

	Διάσταση πλαισίου Four Dimensional Framework			
	Context	**Learner specification**	**Representation**	**Pedagogy**
Παιχνίδι				
Py-rate Adventures	• single-player • προσωπικό ενδιαφέρον • αρχείο οδηγιών	δεν προϋποθέτει εμπειρία στο αντικείμενο και προϋπάρχουσες γνώσεις	δύο διαστάσεων με κινούμενες και στατικές εικόνες	ερωτήσεις πολλαπλής επιλογής
Code Combat	• single-player • προσωπικό ενδιαφέρον • εργαλείο στα πλαίσια μαθήματος • τεχνική υποστήριξη	δεν προϋποθέτει εμπειρία στο αντικείμενο και προϋπάρχουσες γνώσεις	δύο διαστάσεων σε οπτική τρίτου προσώπου	δημιουργία κώδικα στην φιλοσοφία της γλώσσας
Codein Game	• single-player • προσωπικό ενδιαφέρον	προϋποθέτει προγραμματιστική εμπειρία	δύο διαστάσεων με κινούμενες και στατικές εικόνες	δημιουργία λειτουργικού κώδικα
CheckiO	• single-player • προσωπικό ενδιαφέρον • εργαλείο στα πλαίσια μαθήματος	προϋποθέτει προγραμματιστική εμπειρία	χάρτης με νησιά και ελάχιστες κινούμενες εικόνες	δημιουργία λειτουργικού κώδικα

2.5 Συμπεράσματα

Τα παιχνίδια που μελετήθηκαν χρησιμοποιούνται είτε από τον παίκτη από καθαρά προσωπική επιλογή και ενδιαφέρον ενασχόλησης με το αντικείμενο που πραγματεύονται, είτε μπορούν να χρησιμοποιηθούν στα πλαίσια μαθήματος ως εκπαιδευτικό εργαλείο. Στις περισσότερες περιπτώσεις, δεν παρέχεται ιδιαίτερη τεχνική υποστήριξη και αν κάποιος παίκτης συναντήσει προβλήματα λειτουργίας του παιχνιδιού, θα χρειαστεί πιθανώς να τα αντιμετωπίσει μόνος του αν ενδιαφέρεται να συνεχίσει στο παιχνίδι.

Στο πλαίσιο της αναπαράστασης, όλα τα παιχνίδια δεν δίνουν προοπτική και κινούνται στον χώρο των δύο διαστάσεων. Επιπροσθέτως, τα γραφικά είναι, στις περισσότερες των περιπτώσεων, ελάχιστες κινούμενες εικόνες με στατικό περιβάλλον και δεν γίνεται χρήση τρισδιάστατων γραφικών που θα μπορούσε να βοηθήσει στην εμβύθιση (immersion) των παικτών και να τους παροτρύνει μέσω του ψυχαγωγικού κομματιού να συνεχίσουν στο παιχνίδι.

Στη παιδαγωγική διάσταση και τη διάσταση που αφορά στο προφίλ του μαθητή, και συγκεκριμένα για την εκμάθηση της γλώσσας Python, κάθε παιχνίδι έχει δική του λογική και φιλοσοφία όπως αναλύθηκε. Στα παιχνίδια που δεν προϋποθέτουν προγραμματιστική εμπειρία είτε δεν εκτελείτε καθόλου κώδικας Python όπως στο Py-rate Adventures είτε υπάρχει κάποιου είδους προσομοίωση της εκτέλεσης κώδικα όπως στο CodeCombat. Από την άλλη, τα παιχνίδια που προϋποθέτουν προγραμματιστική εμπειρία από τον παίκτη, εκτελούν των κώδικα που δημιουργείται από αυτόν αλλά δεν περιέχουν εισαγωγικά κεφάλαια για την Python ακόμα και στα ευκολότερα επίπεδά τους. Κανένα παιχνίδι, από αυτά που παρουσιάστηκαν, δεν συνδυάζει την δημιουργία και εκτέλεση λειτουργικού κώδικα με την εισαγωγή ενός παίκτη στη γλώσσα προγραμματισμού Python ή ακόμα και στα ευκολότερα επίπεδα των παιχνιδιών που προϋποθέτουν βασικές προγραμματιστικές γνώσεις και εκτελούν κώδικα Python, το οπτικό αποτέλεσμα τελικά προς τον παίκτη είναι ελάχιστο.

3 Μεθοδολογία

3.1 Εισαγωγή

Στα πλαίσια της βιβλιογραφικής ανασκόπησης πραγματοποιήθηκε έρευνα για τον εντοπισμό παιχνιδιών σοβαρού σκοπού που διδάσκουν τον προγραμματισμό με την γλώσσα προγραμματισμού Python και ορίστηκε το θεωρητικό υπόβαθρο. Τα τελικά συμπεράσματα που εξήχθησαν, συμβάλουν στη σχεδίαση και ανάπτυξη του παιχνιδιού που θα δημιουργηθεί στα πλαίσια της διπλωματικής εργασίας.

Το πλαίσιο τεσσάρων διαστάσεων (Four Dimensional Framework) (de Freitas & Oliver, 2006), εκτός από την αξιολόγηση παιχνιδιών σοβαρού σκοπού, χρησιμοποιείται και στη σχεδίαση και δημιουργία παιχνιδιών που ενσωματώνοντας τις πτυχές του μπορούν να επιτύχουν καλύτερα μαθησιακά αποτελέσματα (de Freitas & Jarvis, 2006). Το πλαίσιο αυτό, χρησιμοποιείται στη σχεδίαση του τελικού παιχνιδιού που αναπτύχθηκε στα πλαίσια της διπλωματικής αυτής εργασίας.

3.2 Σκοπός του παιχνιδιού

Σκοπός του παιχνιδιού που θα δημιουργηθεί είναι η εκμάθηση της γλώσσας προγραμματισμού Python. Στόχος είναι να υπάρχει δυνατότητα κάποιος παίκτης με ελάχιστη ή καθόλου προγραμματιστική εμπειρία να μπορεί να είναι σε θέση να παίξει το παιχνίδι και να μην υπάρχουν προαπαιτούμενες γνώσεις. Μέσα από το παιχνίδι και τις δοκιμασίες που θα παρουσιάζονται στον παίκτη, θα πρέπει αυτός μελετώντας το εκπαιδευτικό περιεχόμενο να μπορεί να αντιληφθεί τις βασικές λειτουργίες, συντακτικό, φιλοσοφία και δομή της γλώσσας προγραμματισμού Python. Καθώς θα έχει κατανοήσει όλα τα βασικά τμήματα, θα πρέπει να είναι σε θέση να προχωρήσει σε πιο απαιτητικές δοκιμασίες και ως το τέλος του παιχνιδιού να έχει φτάσει σε πιο προχωρημένο επίπεδο σύνθετων εννοιών στην Python αλλά και στον προγραμματισμό γενικότερα.

Το εκπαιδευτικό περιεχόμενο, πρέπει να συνδέεται άμεσα με το παιχνίδι ή και τον κόσμο στον οποίο διαδραματίζεται με τρόπο που θα υποστηρίζει την κατανόηση των παρουσιαζόμενων αντικειμένων από τους παίκτες. Καθώς γίνεται προσπάθεια σύνδεσης του εκπαιδευτικού με το ψυχαγωγικό μέρος στο παιχνίδι, είναι απαραίτητο τα δύο αυτά μέρη να συνδυάζονται και να ενσωματώνονται στην εμπειρία χρήσης ενός παίκτη και να μην δίνουν το αίσθημα ότι είναι δύο διαφορετικά τμήματα που δεν έχουν σχέση μεταξύ τους.

3.3 Θεωρητική ανάλυση και σχεδίαση παιχνιδιού

Με βάση τις 4 διαστάσεις του Four Dimensional Framework, γίνεται η αρχική σχεδίαση του παιχνιδιού. Θα πρέπει να έχει ξεκαθαριστεί για το παιχνίδι, πριν ξεκινήσει η ανάλυση και η δημιουργία των περιεχομένων του, με ποιον τρόπο θα μπορέσει να ενσωματώσει τις διαστάσεις του πλαισίου σχεδίασης. Επίσης, είναι απαραίτητο να ενσωματωθούν στη σχεδίαση του παιχνιδιού, τα χαρακτηριστικά που εξήχθησαν από τα συμπεράσματα της έρευνας και συγκριτικής αξιολόγησης σχετικών παιχνιδιών σοβαρού σκοπού.

Αρχικά στη διάσταση γενικού πλαισίου (context), το παιχνίδι σχεδιάζεται ως εμπειρία ενός παίκτη (single-player). Κάθε ενδιαφερόμενος που θα θελήσει να παίξει το παιχνίδι, θα έχει λάβει την απόφαση σε προσωπικό επίπεδο είτε για το ψυχαγωγικό μέρος του παιχνιδιού είτε για το εκπαιδευτικό είτε σε συνδυασμό τους που είναι και ο τελικός στόχος. Δεν σχεδιάζεται το συγκεκριμένο παιχνίδι να μπορεί να ενσωματωθεί και να χρησιμοποιηθεί ως εκπαιδευτικό εργαλείο στα πλαίσια μαθήματος ή τάξης και είναι σημαντικό να γνωρίζει κάθε ενδιαφερόμενο μέρος την αρχική σχεδίαση και πεδίο εφαρμογής του.

Ο παίκτης, καθώς θα ασχολείται σε προσωπικό επίπεδο, θα πρέπει να γνωρίζει εξαρχής τον απαραίτητο εξοπλισμό που θα πρέπει να διαθέτει ώστε να παίξει το παιχνίδι. Έτσι, στη σχεδίαση του παιχνιδιού, λαμβάνονται υπόψη τα πιθανά προβλήματα που θα συναντήσει ένας παίκτης και θα γίνει προσπάθεια εξάλειψής τους. Επίσης, καθώς το παιχνίδι δημιουργείται στα πλαίσια εκπόνησης διπλωματικής εργασίας και δεν αναπτύσσεται από κάποια εταιρία ή οργανισμό, η τεχνική υποστήριξη που μπορεί να παρασχεθεί είναι πολύ περιορισμένη. Είναι σημαντικό, το παιχνίδι να αναπτυχθεί σε μία πλατφόρμα βάση της οποίας οι παίκτες θα συναντήσουν τα λιγότερα προβλήματα.

Στη διάσταση που αφορά στον μαθητή (learner specification), έχει ήδη οριστεί ότι το παιχνίδι δεν θα προϋποθέτει κάποιο προγραμματιστικό υπόβαθρο από τον παίκτη και θα αναλαμβάνει να τον εισάγει στο εκπαιδευτικό περιεχόμενο και τις δραστηριότητες που θα περιλαμβάνει. Επίσης, το παιχνίδι θα το έβρισκαν χρήσιμο άτομα που έχουν προγραμματιστική εμπειρία σε κάποια άλλη γλώσσα προγραμματισμού και ενδιαφέρονται να μάθουν την Python. Καθώς στόχος είναι η εκμάθηση της γλώσσας προγραμματισμού Python από εισαγωγικό επίπεδο, το τελικό κοινό στο οποίο απευθύνεται θα μπορούσε να είναι νεότερης ηλικίας άτομα, που πιθανώς δεν έχουν άλλη

επαφή με τον προγραμματισμό, αλλά δεν αποτρέπεται σε καμία περίπτωση και η χρήση του παιχνιδιού από οποιοδήποτε άτομο ενδιαφέρεται ασχέτως ηλικιακής ομάδας.

Σε σχέση με τον τρόπο αναπαράστασης (mode of representation) και την διάσταση αυτή του πλαισίου, σχεδιάζεται το παιχνίδι να έχει υψηλής ποιότητας γραφικά και ως αποτέλεσμα να παρέχει υψηλού επιπέδου πιστότητα (fidelity). Τα γραφικά θα πρέπει να συνδυάζονται και με κάποιο μουσικό υπόβαθρο που δεν θα αποσπά την προσοχή του παίκτη από τις δραστηριότητες που θα συναντήσει και θα συμβάλουν στην μεγαλύτερη εμβύθιση (immersion) των παικτών που οδηγεί σε υψηλότερα ποσοστά συμμετοχής και κίνητρα στους παίκτες να ολοκληρώσουν το εκπαιδευτικό περιεχόμενο.

Σε συνδυασμό με τα παραπάνω, καθώς το παιχνίδι έχει προγραμματιστικό εκπαιδευτικό περιεχόμενο, αυτό θα πρέπει να αντικατοπτρίζεται και στην γραφική σχεδίαση αλλά και στον τρόπο αλληλεπίδρασης του παίκτη με το παιχνίδι. Θα πρέπει σε οποιοδήποτε σημείο και δοκιμασία να είναι σε θέση ο παίκτης να αντιληφθεί πιθανά λάθη και να του δίνεται η κατάλληλη ανατροφοδότηση ώστε να μπορέσει να τα διορθώσει άμεσα. Επίσης, η εκτέλεση κώδικα που δίνει λύση στο πρόβλημα που παρουσιάζεται, μπορεί να έχει αποτελέσματα που σχετίζονται με το θέμα του προβλήματος όπου είναι εφικτό, στον κόσμο του παιχνιδιού ή τον κεντρικό χαρακτήρα και γίνονται άμεσα αντιληπτά και κατανοητά από τον παίκτη. Έτσι, επιδιώκεται ένα επιπλέον σύστημα εκμάθησης και κατανόησης του αντικειμένου που μελετάται, εκτός της μελέτης της θεωρίας και της σωστής εφαρμογής της.

Σε αυτό το σημείο, πρέπει να συμπεριληφθούν στην σχεδίαση και τα χαρακτηριστικά του παιχνιδιού (gameplay) ώστε να γίνεται το ψυχαγωγικό μέρος του παιχνιδιού περισσότερο διασκεδαστικό και να μην αποτελεί εμπόδιο στην εκπαιδευτική διαδικασία. Ο χειρισμός του χαρακτήρα θα πρέπει να είναι ακριβής και σαφής καθώς και η λογική στο σύστημα του παιχνιδιού ως σύνολο. Είναι απαραίτητο, η διαδικασία του ψυχαγωγικού μέρους να είναι ευχάριστη και να μην αποτελεί εμπόδιο στη διδασκαλία του εκπαιδευτικού περιεχομένου.

Τέλος, στη παιδαγωγική διάσταση του πλαισίου σχεδίασης (pedagogy) πρέπει να οριστεί το εκπαιδευτικό περιεχόμενο και ο τρόπος παρουσίασής του. Καθώς το παιχνίδι θα αναλαμβάνει να εισάγει τον παίκτη, ακόμα και αν αυτός δεν έχει καμία προηγούμενη εμπειρία με τον προγραμματισμό, είναι απαραίτητο να δίνονται σαφείς οδηγίες και το εκπαιδευτικό περιεχόμενο να επεξηγείται με τρόπο κατανοητό για κάθε κατηγορία πιθανών παικτών. Επίσης, θα πρέπει το τελικό παιχνίδι να οδηγεί τον παίκτη στη

δημιουργία κώδικα που θα είχε άμεση χρήση σε πραγματικό περιβάλλον προγραμματισμού και να μπορεί να παρέχει στον παίκτη δυνατότητα εκτέλεσης κώδικα που αυτός θα δημιουργεί. Έτσι, ο παίκτης θα είναι σε θέση να δει και να μάθει το κομμάτι της εκτέλεσης εντολών σε Python και τι προβλήματα και ιδιαιτερότητες θα συναντήσει κατά τη χρήση της γλώσσας. Ταυτόχρονα, είναι σημαντικό ο παίκτης να προστατεύεται και να καθοδηγείται στις περιπτώσεις που συναντάει δυσκολίες επίλυσης δοκιμασιών και γρίφων που ενσωματώνονται ώστε να αποτρέπεται η πρόωρη εγκατάλειψη του παιχνιδιού.

Το εκπαιδευτικό περιεχόμενο θα καλύπτει τα βασικά τμήματα της γλώσσας και θα οδηγεί τους παίκτες βαθμιαία σε πιο σύνθετα. Φτάνοντας στα τελευταία επίπεδα του παιχνιδιού, θα πρέπει ένας παίκτης που έχει κατανοήσει την ύλη που συμπεριλαμβάνεται σε προηγούμενα επίπεδα να είναι σε θέση να αναγνωρίσει και να χρησιμοποιήσει τη γνώση που έλαβε και να την συνδυάσει με το νέο υλικό που του παρουσιάζεται ώστε να λύσει το πρόβλημα της κάθε δοκιμασίας.

3.4 Τεχνολογίες που χρησιμοποιήθηκαν

Βάσει της θεωρητικής ανάλυσης και σχεδίασης που πραγματοποιήθηκε στο κεφάλαιο 3.3, έπρεπε να ληφθούν αρχικά συγκεκριμένες αποφάσεις που θα καθορίσουν και την πορεία υλοποίησής του. Λόγω της πολυπλοκότητας του περιεχομένου αλλά και το γεγονός ότι η δημιουργία και ανάπτυξη γίνεται από ένα άτομο, αποφασίστηκε η χρήση κάποιας μηχανής παιχνιδιών. Σύμφωνα με τον Halpern (2018) οι μηχανές παιχνιδιών είναι εργαλεία ανάπτυξης λογισμικού σχεδιασμένα ώστε να μειώσουν το κόστος, πολυπλοκότητα και χρόνο που απαιτείται για την ανάπτυξη και τελική υλοποίηση ηλεκτρονικών παιχνιδιών.

Οι μηχανές παιχνιδιών παρέχουν έτοιμη την λειτουργικότητα, συστήματα γραφικών, συστήματα διαχείρισης ήχου και συστήματα τεχνητής νοημοσύνης (Jain, 2020). Έτσι, υπάρχει δυνατότητα εστίασης στα χαρακτηριστικά και λειτουργίες που πρέπει να περιλαμβάνονται στο τελικό παιχνίδι που αναπτύσσεται στα πλαίσια και βάσει της εργασίας, χωρίς να υπάρχουν προβληματισμοί έρευνας και ανάπτυξης της βασικής λειτουργικότητας του παιχνιδιού σε επίπεδο εφαρμογής.

Αρχικά, εξετάστηκε το ενδεχόμενο δημιουργίας του παιχνιδιού μόνο με χρήση Python καθώς δίνει την δυνατότητα σύνδεσης του κώδικα που διδάσκεται και δημιουργεί ο παίκτης με την πραγματική εκτέλεσή του.

Το ενδεχόμενο αυτό όμως σύντομα απορρίφθηκε καθώς πολλά από τα πακέτα Python που λειτουργούν ως μηχανές και πλαίσια ανάπτυξης παιχνιδιών, αδυνατούν να δημιουργήσουν σύνθετα παιχνίδια υψηλών γραφικών προδιαγραφών που θα ενσωματώνουν τα στοιχεία που κρίνονται απαραίτητα στην ανάλυση και σχεδίαση που πραγματοποιήθηκε.

Βάσει αυτής της ανάλυσης, ήταν αναγκαία μια μηχανή παιχνιδιών που θα συνδυάζει τη δυνατότητα δημιουργίας σύνθετων γραφικά παιχνιδιών με υψηλά επίπεδα πιστότητας αλλά και την ενσωμάτωση του μέρους κώδικα Python που θα διδάσκει το παιχνίδι. Έτσι, επιλέχθηκε η χρήση της μηχανής παιχνιδιών Unreal Engine.

3.4.1 Unreal Engine 4

Η μηχανή παιχνιδιών Unreal Engine 4 (Unreal Engine, n.d.) παρέχει ένα πλήρες σύστημα ανάπτυξης που ενσωματώνει σύγχρονες τεχνολογίες (Unreal Engine | Features, n.d.). Εκτός των βασικών ιδιοτήτων των μηχανών παιχνιδιών, κρίνονται απαραίτητα για την ανάπτυξη του παιχνιδιού βάσει της ανάλυσης που πραγματοποιήθηκε ορισμένα από τα χαρακτηριστικά που υποστηρίζονται σύμφωνα με (Unreal Engine | Features, n.d.):

- Αυτοματοποίηση και δημιουργία περιβάλλοντος και κόσμου παιχνιδιού καθώς και επεξεργασίας και επέκτασής του
- Βελτιστοποίηση πόρων και στοιχείων που χρησιμοποιούνται
- Οικοσύστημα διάθεσης και διανομής υψηλής ποιότητας και πιστότητας γραφικών και επιμέρους λειτουργιών
- Υποστήριξη τελικής εφαρμογής σε πληθώρα συστημάτων που μπορεί να γίνει χρήση της
- Ανοιχτή πρόσβαση στον πηγαίο κώδικα σε C++ με δυνατότητες προσαρμογής και επέκτασης που ταιριάζει στο παιχνίδι που δημιουργείται

3.4.2 Χρήση Python στο gameplay

Βάσει του τελευταίου χαρακτηριστικού από τις δυνατότητες της Unreal Engine 4 που κρίνονται σημαντικές και παρουσιάστηκαν, υπάρχει εφικτός τρόπος επέκτασης της λειτουργικότητας προγραμμάτων C++ με χρήση Python που κρίνεται σημαντική για το παιχνίδι. Η Python παρέχει πακέτα ενσωμάτωσης (embedding) και επέκτασης σε εφαρμογές C ή C++ ώστε αυτές να έχουν πρόσβαση και να μπορούν να χρησιμοποιούν

δυνατότητές της κατά την εκτέλεση (Embedding Python in Another Application — Python 3.9.1 documentation, n.d.).

Σε αυτό το σημείο, πρέπει να σημειωθεί ότι η εισαγωγή τέτοιου είδους στοιχείων στην εφαρμογή που δεν αποτελούν κομμάτι της μηχανής παιχνιδιών που επιλέχθηκε ενέχει το κίνδυνο δημιουργίας και εμφάνισης διαφόρων προβλημάτων στο τελικό παιχνίδι. Έτσι, πρέπει να κατασκευαστεί ένα σύστημα που θα επιτρέπει την επέκταση της C++ που χρησιμοποιεί στον πυρήνα της η Unreal Engine 4 αλλά δεν θα καταστρέφει την λειτουργικότητα της εφαρμογής.

Ταυτόχρονα, θα πρέπει να δημιουργηθεί δυνατότητα κλήσης και χρήσης Python από την λειτουργία της μηχανής, η οποία θα είναι σε θέση να διαχειριστεί τον κώδικα που θα δημιουργεί και θα εισάγει ο παίκτης. Για να συμβεί αυτό, θα πρέπει να υπάρχει επικοινωνία της C++ που χρησιμοποιείται από τη μηχανή με την Python κατά την εκτέλεση και λειτουργία του παιχνιδιού.

4 Ανάλυση και σχεδίαση

4.1 Περιγραφή παιχνιδιού

Το παιχνίδι σοβαρού σκοπού που δημιουργήθηκε και παρουσιάζεται σε αυτό και τα επόμενα κεφάλαια έχει τον τίτλο «Journey of the PyVenturer». Είναι ένα παιχνίδι τύπου single-player, που εντάσσεται στην κατηγορία παιχνιδιών action role - playing (Action RPG) καθώς εμπεριέχει στοιχεία και των δύο κλάσεων κατηγοριοποίησης παιχνιδιών. Ο τίτλος σχετίζεται με το εκπαιδευτικό περιεχόμενο, εκμάθησης της γλώσσας προγραμματισμού Python, και το ψυχαγωγικό μέρος που είναι το ταξίδι (Journey) του κεντρικού χαρακτήρα (PyVenturer) με τις διάφορες δοκιμασίες που καλείται να επιλύσει εκπαιδευτικού, στη γλώσσα προγραμματισμού Python, και ψυχαγωγικού τύπου.

Ο παίκτης, που βρίσκεται στην θέση του κεντρικού χαρακτήρα-ήρωα του παιχνιδιού, έχει την οπτική τρίτου προσώπου (third person). Κατά τη διάρκεια του ταξιδιού (Journey), ο ήρωας συναντάει διάφορες δοκιμασίες που πρέπει να ξεπεράσει. Αυτές οι δοκιμασίες σχετίζονται με κατανόηση θεωρίας Python που παρουσιάζεται στον παίκτη και πρέπει αυτός να χρησιμοποιήσει στην επίλυση του προβλήματος, οπού κατά τη σωστή απάντηση θα του παρέχονται τα εργαλεία και οι δυνατότητες εντός παιχνιδιού να ανταπεξέλθει στις δυσκολίες του gameplay. Αυτές οι δυσκολίες έχουν την μορφή εχθρών όπου η δυναμική τους μεγαλώνει με κάθε νέο επίπεδο στο παιχνίδι.

Το εκπαιδευτικό περιεχόμενο βρίσκεται στα διάφορα σημεία ενδιαφέροντος (checkpoints) που είναι διασκορπισμένα στον κόσμο του παιχνιδιού. Μόλις ο ήρωας φτάσει σε κάποιο από αυτά τα σημεία, εμφανίζεται στην οθόνη του το πλαίσιο της θεωρίας όπου γίνεται επεξήγησή της και δίνονται παραδείγματα. Στο τέλος της θεωρίας δίνονται στον παίκτη οδηγίες και διευκρινίσεις σχετικά με τη δοκιμασία που καλείται να λύσει. Ακριβώς κάτω από το μέρος της θεωρίας, εμφανίζεται ένας τύπος κονσόλας (console) που μπορεί κανείς να συναντήσει σε παρόμοια μορφή σε κάποιο περιβάλλον ανάπτυξης. Ο παίκτης συμπληρώνει και τελικά εκτελεί τον κώδικα που είναι απαραίτητος στο πρόβλημα που του παρουσιάζεται κάθε φορά. Να σημειωθεί, ότι παρόλο που γίνεται εισαγωγή και εκτέλεση κώδικα Python, δεν υπάρχει πεδίο ή μηχανισμός εξόδου και προβολής της λειτουργίας του κώδικα που εκτελέστηκε αλλά σε περίπτωση που ο παίκτης έχει βρει τη σωστή λύση, τα αποτελέσματα είναι άμεσα φανερά και αντιληπτά στο περιβάλλον και τον κόσμο του παιχνιδιού. Σε αντίθετη

περίπτωση, όπου ο παίκτης έκανε λάθος είτε δημιουργίας έγκυρου κώδικα Python είτε λάθος σχετικό με το πρόβλημα που του παρουσιάζεται, του δίνεται κάθε φόρα η απαραίτητη ανατροφοδότηση (feedback) από το παιχνίδι ώστε να το διορθώσει.

4.2 Εκπαιδευτικό περιεχόμενο

Το παιχνίδι σοβαρού σκοπού που δημιουργείται, όπως προαναφέρθηκε, στοχεύει στην εισαγωγή των παικτών στη γλώσσα προγραμματισμού Python και την εκμάθησή μέχρι και σύνθετων εννοιών και δομών. Επίσης, δεν προϋποθέτει προγραμματιστική εμπειρία από τον παίκτη ακόμα και αν αυτή υπάρχει. Συνδυάζοντας τα παραπάνω, το εκπαιδευτικό περιεχόμενο είναι σχεδιασμένο με τρόπο που να τα ενσωματώνει. Παρόλο που τα επιμέρους θέματα και έννοιες της Python είναι χωρισμένα στα διάφορα επίπεδα του παιχνιδιού, δεν υπάρχει κάποιος ιδιαίτερος τρόπος που διαχωρίζονται ως εννοιολογικά κεφάλαια. Αντί αυτού, καθώς το παιχνίδι είναι εμπειρία ενός παίκτη (single-player experience), έμφαση δίνεται στην ενσωμάτωση της διδακτικής ύλης στην ιστορία, την εξέλιξη και την πρόοδο του παίκτη στο παιχνίδι αλλά και στην βαθμιαία αύξηση της πολυπλοκότητας των θεμάτων που αναλύονται και παρουσιάζονται. Απώτερος στόχος είναι η διδασκαλία της γλώσσας στον παίκτη και όχι τόσο η δημιουργία ενός παιχνιδιού που θα μπορούσε να ενταχθεί στα πλαίσια συμπλήρωσης της διδακτέας ύλης κάποιου μαθήματος.

Επιπλέον, τα επίπεδα και το εκπαιδευτικό περιεχόμενο που περιέχουν, αν θεωρηθούν ως κεφάλαια στη διδασκαλία της Python, συνδέονται άμεσα με τις καταστάσεις και τα γεγονότα εντός παιχνιδιού και κατά αυτόν τον τρόπο γίνεται ο διαχωρισμός τους στα επιμέρους επίπεδα. Τα προβλήματα και οι ενότητες διδασκαλίας που περιλαμβάνει το παιχνίδι σε κάθε επίπεδο παρουσιάζονται στον Πίνακα 2.

Πίνακας 2: Ενότητες επιπέδων και το εκπαιδευτικό περιεχόμενο που περιέχουν

Επίπεδο	Ενότητες επιπέδου	Εκπαιδευτικό περιεχόμενο
Επίπεδο 1	1. Print Strings	▪ Χρήση της εντολής print( ) ▪ Χρήση String στην Python
	2. Integer Variables	▪ Έννοια και χαρακτηριστικά μεταβλητών και ακέραιων μεταβλητών

		▪ Χρήση μεταβλητών
	3. Setting Integer Variables	▪ Ανάθεση ακέραιων τιμών σε μεταβλητές
	4. Floating Point Variables	▪ Έννοια και χαρακτηριστικά μεταβλητών με τιμές με δεκαδικό μέρος ▪ Ανάθεση τιμών με δεκαδικό μέρος σε μεταβλητές
Επίπεδο 2	1. Basic Operators	▪ Χρήση βασικών συμβόλων πράξεων
	2. String Formatting	▪ Έννοια και χρήση μορφοποίησης String
	3. Lists	▪ Έννοια και χαρακτηριστικά δομής δεδομένων λίστας ▪ Ανάθεση περιεχομένου και χρήση λίστας
Επίπεδο 3	1. Conditions - If / elif /else statements	▪ Έννοια και χρήση λογικών συνθηκών ▪ Τελεστές σύγκρισης λογικών συνθηκών ▪ Έννοια και λειτουργία δομής if / elif / else ▪ Χρήση δομής if / elif / else
	2. While Loops	▪ Έννοια και λειτουργία δομής while ▪ Χρήση δομής while
	3. Indexing	▪ Έννοια θέσης σε δομές δεδομένων ▪ Χρήση θέσης και ανάκτηση δεδομένων από δομές δεδομένων

	4. Slicing	▪ Έννοια ανάκτησης υποσυνόλου από δομή δεδομένων ▪ Χρήση ανάκτησης υποσυνόλου
Επίπεδο 4	1. For Loops	▪ Έννοια και λειτουργία δομής for ▪ Χρήση δομής for
	2. Dictionaries	▪ Έννοια και χαρακτηριστικά δομής δεδομένων dictionary ▪ Ανάθεση περιεχομένου και χρήση dictionaries
	3. Functions	▪ Έννοια και χαρακτηριστικά συναρτήσεων ▪ Λειτουργία και χρήση συναρτήσεων ▪ Δημιουργία και χρήση συναρτήσεων
Επίπεδο 5	1. Object Oriented Programming (OOP)	▪ Έννοια και χαρακτηριστικά αντικειμενοστρεφούς προγραμματισμού ▪ Λειτουργία και χρήση κλάσεων και αντικειμένων ▪ Δημιουργία και χρήση κλάσεων και αντικειμένων
	2. OOP-Inheritance	▪ Έννοια και χαρακτηριστικά κληρονομικότητας στον αντικειμενοστρεφή προγραμματισμό ▪ Λειτουργία και χρήση αντικειμένων που κληρονομούν ▪ Δημιουργία και χρήση αντικειμένων που κληρονομούν
	3. Try/Except	▪ Έννοια και λειτουργία δομής

		try / except ▪ Χρήση δομής try / except
	4. Modules	▪ Έννοια και χαρακτηριστικά module ▪ Λειτουργία και χρήση module

Κάθε ενότητα των επιμέρους επιπέδων αντιστοιχεί και σε μια δοκιμασία εντός του παιχνιδιού την οποία καλείται ο παίκτης να επιλύσει, βασιζόμενος στη γνώση της θεωρίας από προηγούμενες δοκιμασίες, όπου είναι απαραίτητο, και εφαρμογή του παρουσιαζόμενου αντικειμένου.

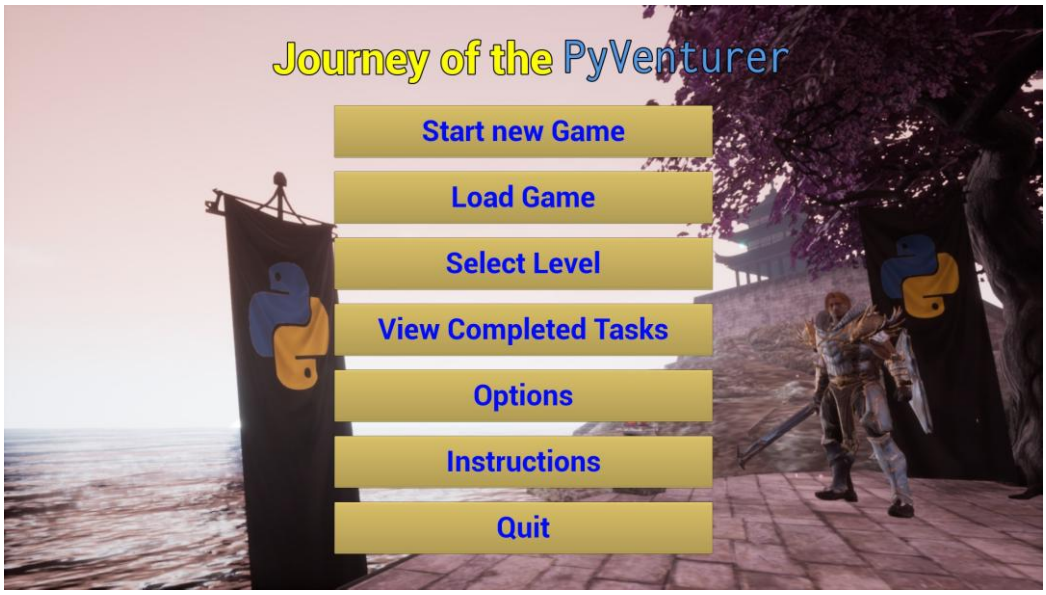
4.3 User Interface του παιχνιδιού

Με την εκτέλεση του παιχνιδιού, ο παίκτης δεν εισέρχεται αμέσως στο gameplay. Αντί αυτού, εμφανίζεται στην οθόνη του το κεντρικό μενού (Main Menu) του παιχνιδιού μέσα από το οποίο έχει τη δυνατότητα ο κάθε παίκτης να κάνει επιλογές σε σχέση με τη λειτουργία αλλά και τις δραστηριότητες του παιχνιδιού.

Εικόνα 4: Αρχική οθόνη και κεντρικό μενού στο Journey of the PyVenturer

Μέσα από αυτό το μενού ο χρήστης με την πρώτη επιλογή μπορεί να ξεκινήσει άμεσα ένα νέο παιχνίδι (επιλογή “Start new Game”), όπου και οδηγείται στον αρχικό κόσμο του παιχνιδιού. Οι επόμενες δύο επιλογές “Load Game” και “Select Level” είναι απενεργοποιημένες για έναν παίκτη που εισέρχεται για πρώτη φορά στο παιχνίδι. Αυτές

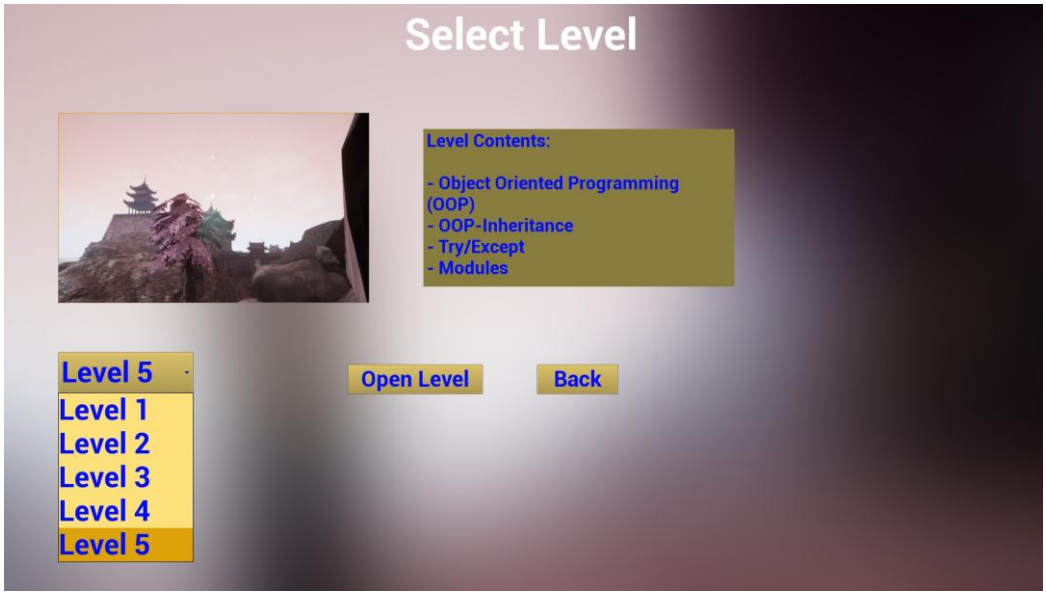
οι επιλογές θα ενεργοποιηθούν μόλις αυτός ξεκινήσει να παίζει το παιχνίδι και λύσει κάποιες αρχικές δοκιμασίες. Το κεντρικό μενού μετατρέπεται όπως φαίνεται στην Εικόνα 5 με δυνατότητα επιλογής των προηγουμένως απενεργοποιημένων επιλογών.



Εικόνα 5: Αρχική οθόνη και κεντρικό μενού με όλες τις δυνατότητες

Αφού πλέον ο παίκτης έχει ενεργοποιήσει όλες τις δυνατές επιλογές, βάσει του παιχνιδιού και της πορείας του σε αυτό, μπορεί να επιλέξει την επιλογή “Load Game”, η οποία θα τον επαναφέρει στο επίπεδο που άφησε το παιχνίδι την τελευταία φορά. Πρέπει να σημειωθεί ότι αυτή η επιλογή είναι κυριολεκτική, υπό την έννοια ότι ακόμα και αν κάποιος παίκτης έχει προχωρήσει σε επόμενα επίπεδα αλλά αποφάσισε να επανέρθει σε προηγούμενο επίπεδο για επανάληψη στα διδασκόμενα θέματα κάποιου επιπέδου και αποχώρησε από το παιχνίδι σε αυτό το επίπεδο, η επιλογή “Load Game” θα τον επαναφέρει στο επίπεδο που βρισκόταν πριν αποχωρήσει.

Ο παίκτης όμως, έχει τη δυνατότητα να χρησιμοποιήσει από το κεντρικό μενού τη λειτουργία “Select Level”. Εδώ αφαιρείται από την οθόνη το κεντρικό μενού και δημιουργείται ένα άλλο στο οποίο υπάρχει δυνατότητα επιλογής επιπέδου που έχει καταφέρει ο παίκτης να κερδίσει. Έτσι, ακόμα και αν “ έχασε ” κάποιος παίκτης το τελευταίο σημείο στο μεγαλύτερο επίπεδο που είχε φτάσει από την επιλογή “Load Game”, του παρέχεται η δυνατότητα επαναφοράς από το μενού “Select Level”. Το μενού που δημιουργείται, παρουσιάζεται στην Εικόνα 6.

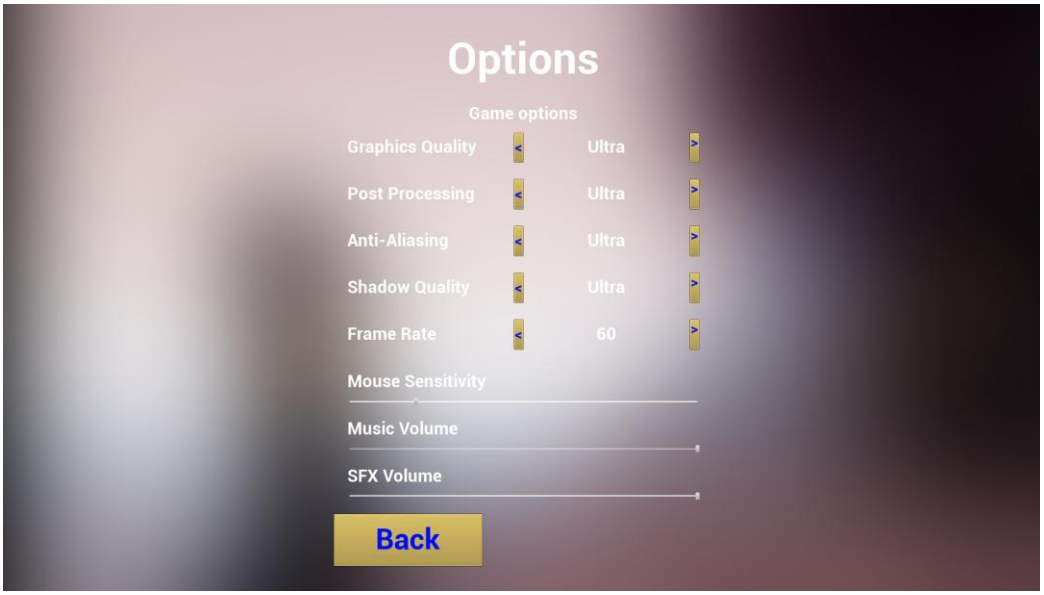


Εικόνα 6: Μενού επιλογής επιπέδου "Select Level"

Σε αυτή την οθόνη μενού, εκτός από μια λίστα με όλα τα επίπεδα που έχει νικήσει ο παίκτης, υπάρχει και ένας πίνακας με το εκπαιδευτικό περιεχόμενο των δραστηριοτήτων που περιλαμβάνει το κάθε επίπεδο. Με την επιλογή, του παίκτη, "Open Level" ξεκινάει το παιχνίδι στο επίπεδο που διάλεξε από τη λίστα με τα διαθέσιμα επίπεδα. Με την επιλογή "Back", επιστρέφει στην οθόνη του κεντρικού μενού.

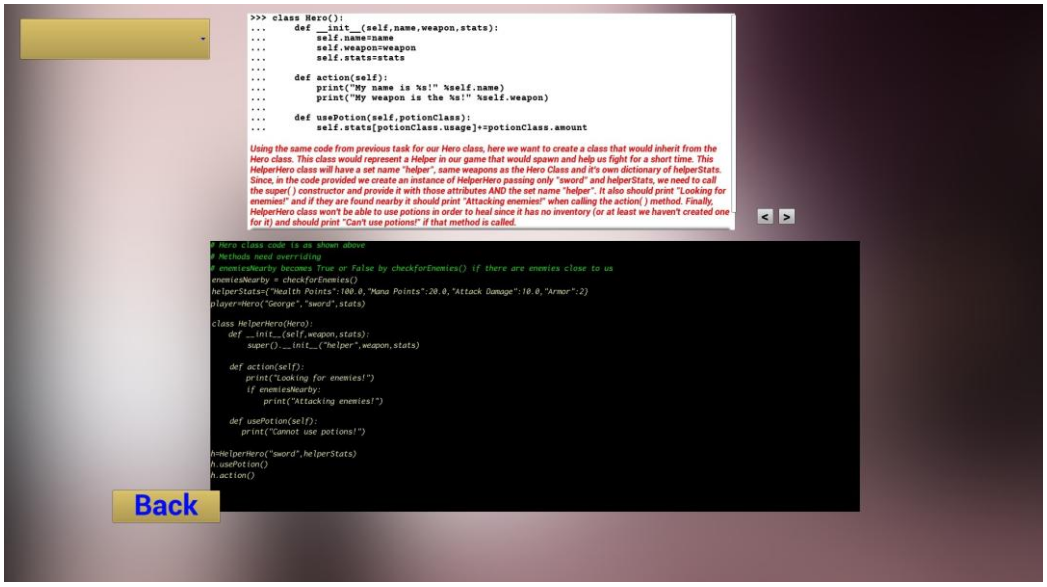
Επίσης, στο κεντρικό μενού υπάρχουν και οι επιλογές "Options", "Instructions" και "Quit" που οδηγούν τον παίκτη σε οθόνες μενού σχετικά με τις διαθέσιμες επιλογές τεχνικών θεμάτων του παιχνιδιού, οδηγίες σχετικά με το gameplay και έξοδο από το παιχνίδι αντίστοιχα.

Στο μενού "Options", ο παίκτης έχει την δυνατότητα να μεταβάλλει τις βασικές ρυθμίσεις του παιχνιδιού ώστε να ταιριάζουν στις προτιμήσεις του και τη δυναμική του συστήματος που διαθέτει. Οι διαθέσιμες επιλογές παρουσιάζονται στην Εικόνα 7 και σχετίζονται με θέματα επιπέδου γραφικών, ταχύτητας κίνησης της κάμερας χρησιμοποιώντας το ποντίκι και ελέγχοντας την ευαισθησία του αλλά και επιπέδων ήχου.



Εικόνα 7: Μενού επιλογής ρυθμίσεων "Options"

Τέλος, στο κεντρικό μενού υπάρχει η επιλογή "View Completed Tasks" η οποία δίνει τη δυνατότητα στον παίκτη να ξαναδεί τις ασκήσεις που περιλαμβάνουν οι δοκιμασίες με τις απαντήσεις έτοιμες, σε περίπτωση που χρειάζεται να επαληθεύσει ή να ξαναδιαβάσει μέρος της θεωρίας χωρίς να παίξει στο παιχνίδι.

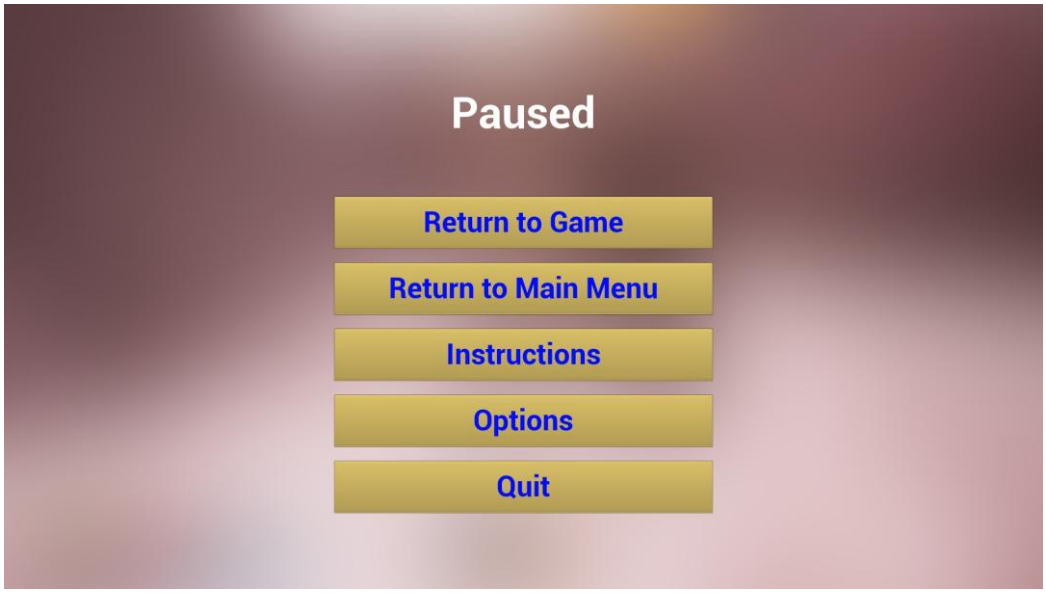


Εικόνα 8: Μενού ολοκληρωμένων δοκιμασιών "View Completed Tasks"

Στο μενού αυτό, όπως και στα μενού επιλογών ρυθμίσεων και επιπέδου, υπάρχει η επιλογή "Back" που επιστρέφει τον παίκτη στην αρχική οθόνη του κεντρικού μενού και μια λίστα στο επάνω και αριστερό μέρος η οποία περιέχει τις δοκιμασίες που ο

παίκτης έχει ολοκληρώσει. Το υπόλοιπο τμήμα της οθόνης, όπως φαίνεται στην Εικόνα 8, περιέχει τη δοκιμασία λυμένη με τον τρόπο που παρουσιάζεται στον παίκτη εντός παιχνιδιού και θα γίνει ανάλυσή τους σε επόμενα κεφάλαια.

Κατά τη διάρκεια του παιχνιδιού, ο παίκτης μπορεί με το πάτημα του κατάλληλου πλήκτρου να σταματήσει την εκτέλεση και ροή του παιχνιδιού (Pause) για όσο χρόνο επιθυμεί και του παρέχονται επιλογές παρόμοιες με το αρχικό μενού.



Εικόνα 9: Μενού "Paused" του παιχνιδιού

4.4 Τεχνικά θέματα

Το παιχνίδι, δοκιμάστηκε και διερευνήθηκε η συμπεριφορά της εφαρμογής σε διάφορα συστήματα, με ποικίλες δυνατότητες και δυναμική και δεν παρατηρήθηκαν σοβαρά ζητήματα προαπαιτούμενων σε αυτό το κομμάτι. Αρχικά, είναι βασική προϋπόθεση να χρησιμοποιείται σύστημα αρχιτεκτονικής 64-bit με λειτουργικό Windows, τουλάχιστον Windows 7 SP1. Επίσης, σε περίπτωση που το σύστημα δεν διαθέτει τα απαραίτητα προγράμματα οδήγησης που απαιτεί μια εφαρμογή Unreal Engine 4, εκτελείται αυτόματα κατά την πρώτη εκκίνηση του παιχνιδιού πρόγραμμα εγκατάστασής τους, που παρέχεται από την μηχανή παιχνιδιών Unreal Engine 4.

5 Παρουσίαση του παιχνιδιού

5.1 Επίπεδα παιχνιδιού

Το παιχνίδι σοβαρού σκοπού «Journey of the PyVenturer» απαρτίζεται από πέντε διαφορετικά επίπεδα. Το περιβάλλον και ο κόσμος του παιχνιδιού σε κάθε επίπεδο είναι διακριτός από τα προηγούμενα αλλά ταυτόχρονα έχει γίνει προσπάθεια σύνδεσης των επίπεδων με τα κατάλληλα γραφικά στοιχεία (textures) ώστε να δίνεται μια αίσθηση συνέχειας στον παίκτη. Συγκεκριμένα, η αρχή (με εξαίρεση το πρώτο επίπεδο) και το τέλος κάθε επιπέδου έχουν θεματολογία και γραφικά που ταιριάζουν και το συνδέουν με το προηγούμενο και το επόμενο επίπεδο.

Τα περιεχόμενα που παρουσιάζονται σε κάθε επίπεδο όσον αφορά τόσο σε θεματολογία όσο και σε εκπαιδευτικό υλικό, όπως προαναφέρθηκε, έγινε προσπάθεια να ενταχθούν στον κόσμο του παιχνιδιού. Έτσι, η σωστή επίλυση των προβλημάτων εκτός από την πρόοδο του παίκτη στο παιχνίδι, έχει και οπτικό αποτέλεσμα άμεσα αναγνωρίσιμο από τον παίκτη ώστε να μπορεί να είναι σε θέση να αντιληφθεί καλύτερα τη διδασκόμενη θεματολογία.

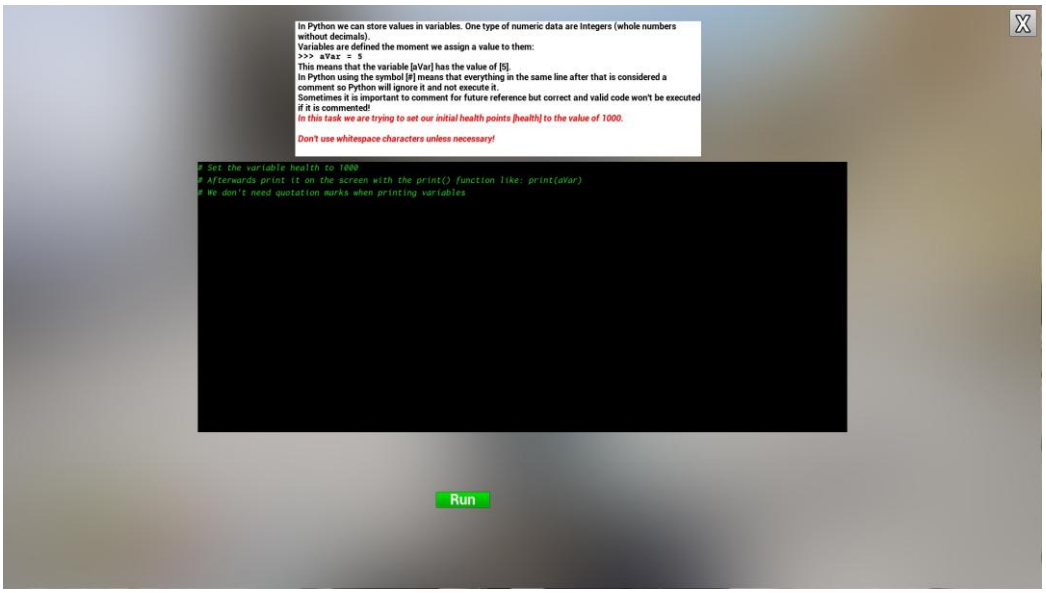
5.1.1 Επίπεδο 1

Εικόνα 10: Περιβάλλον επιπέδου 1

Κατά την διαδρομή του παίκτη στο επίπεδο 1 αλλά και σε κάθε επίπεδο του παιχνιδιού, βρίσκονται τοποθετημένα διάφορα στοιχεία ενδιαφέροντος (checkpoints). Τα σημεία έχουν τη μορφή που παρουσιάζεται στην Εικόνα 11.

Εικόνα 11: Σημείο ενδιαφέροντος (checkpoint) - Επίπεδο 1

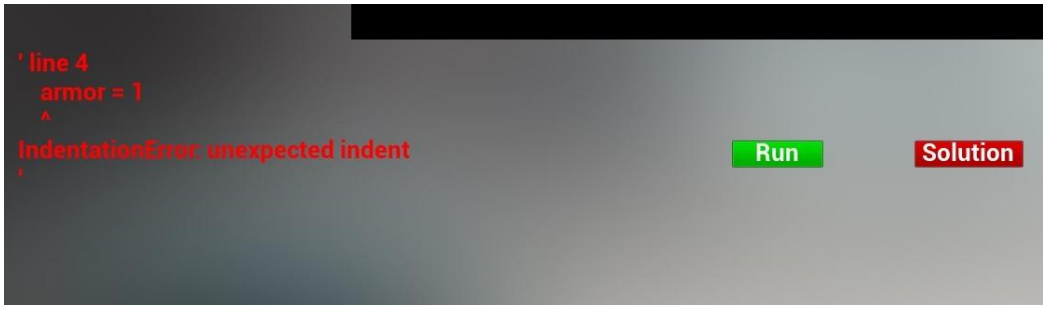
Ο παίκτης πρέπει να βρει όλα τα σημεία ενδιαφέροντος σε κάθε επίπεδο και να φέρει τον κεντρικό χαρακτήρα - ήρωα του παιχνιδιού, τον οποίο χειρίζεται, εντός κάθε σημείου. Τότε, εμφανίζεται στην οθόνη του παίκτη η δοκιμασία όπως φαίνεται στην Εικόνα 12.



Εικόνα 12: Δοκιμασία 2 - Επίπεδο 1

Αρχικά, στην οθόνη παρουσιάζεται ένα πλαίσιο που περιλαμβάνει:

- Στο επάνω μέρος, υπάρχει το πλαίσιο θεωρίας και επεξήγησής της, καθώς και η εκφώνηση του προβλήματος το οποίο καλείται να λύσει ο παίκτης (με κόκκινα γράμματα).
- Στο κεντρικό μέρος της οθόνης, εμφανίζεται το πλαίσιο συμπλήρωσης κώδικα, σε μορφή κονσόλας όπως σε ένα περιβάλλον ανάπτυξης, όπου ο παίκτης θα δημιουργήσει τον κώδικά του.
- Στο κάτω μέρος της οθόνης υπάρχει επιλογή “Run” για την τελική εκτέλεση και έλεγχο του κώδικα που συμπληρώθηκε από τον παίκτη.
- Επάνω και δεξιά στην οθόνη, υπάρχει επιλογή για κλείσιμο του παράθυρου της δοκιμασίας που μπορεί να πατήσει ο παίκτης χωρίς κάποια συνέπεια.
- Τέλος, μετά από πολλαπλές συνεχόμενες λανθασμένες απαντήσεις από τον παίκτη εμφανίζεται αυτόματα το κουμπί επίλυσης της άσκησης “Solution” όπως φαίνεται στην Εικόνα 13. Επιλέγοντάς το, συμπληρώνεται αυτόματα ο απαιτούμενος κώδικας για τον παίκτη.

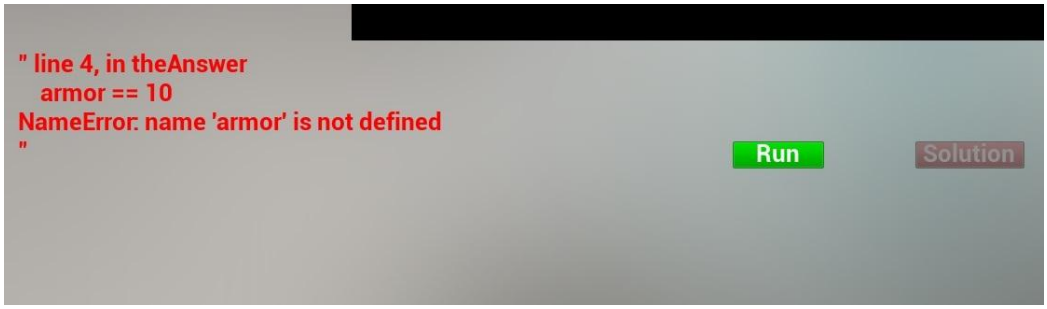


Εικόνα 13: Δοκιμασία 4 - Επίπεδο 1 - Πολλαπλές λανθασμένες απαντήσεις και εμφάνιση επιλογής “Solution”

Με βάση το λάθος που μπορεί να έχει κάνει κάποιος παίκτης, εμφανίζεται επίσης στο κάτω και αριστερό μέρος της οθόνης μήνυμα σε σχέση με το λάθος και το είδος του. Με αυτόν τον τρόπο, δίνεται συνεχώς στον παίκτη η κατάλληλη ανατροφοδότηση (feedback) ώστε να προσπαθήσει να λύσει τη δοκιμασία.

Πρέπει να σημειωθεί ότι ο παίκτης έχει συγκεκριμένο αριθμό βοηθειών που μπορεί να χρησιμοποιήσει, κατά τη διάρκεια του παιχνιδιού. Σε περίπτωση που επιλέξει να χρησιμοποιήσει την επιλογή “Solution”, χάνεται μία από τις διαθέσιμες λύσεις που

έχει. Οι διαθέσιμες, βοήθειες για τον παίκτη, φαίνονται επάνω και δεξιά κατά τη ροή του παιχνιδιού, όπως στις Εικόνες 10 και 11. Επίσης, σε περίπτωση που στον παίκτη δεν έχουν απομείνει άλλες διαθέσιμες βοήθειες, το κουμπί "Solution" είναι απενεργοποιημένο, όπως φαίνεται στην Εικόνα 14.



Εικόνα 14: Δοκιμασία 4 - Επίπεδο 1 - Πολλαπλές λανθασμένες απαντήσεις και εμφάνιση επιλογής "Solution" απενεργοποιημένη

Μετά την σωστή απάντηση, σε κάθε πρόβλημα που παρουσιάζεται, το σημείο ενδιαφέροντος αφαιρείται από τον κόσμο του παιχνιδιού. Σε περίπτωση που ο παίκτης αποφασίσει να κλείσει την δοκιμασία χωρίς να έχει λύσει το πρόβλημα το σημείο παραμένει ως έχει και πρέπει να το επισκεφθεί μελλοντικά ώστε να δώσει την λύση.

Όπως αναφέρθηκε, ο κώδικας που δημιουργεί ο παίκτης σε κάθε επίπεδο και κάθε δοκιμασία έχει αποτελέσματα άμεσα αντιληπτά στον κόσμο του παιχνιδιού και στον κεντρικό χαρακτήρα και σχετίζονται με την λειτουργία του κώδικα σε Python που εισήγαγε.

Αρχικά, ο παίκτης καλείται να τυπώσει με χρήση της εντολής `print()` από την γλώσσα Python, το όνομα του κεντρικού χαρακτήρα - ήρωα. Ότι όνομα επιλέξει ο παίκτης, αφού η εντολή είναι σωστή συντακτικά για την Python, θα εμφανιστεί επάνω από τον ήρωα.

Μετά την επίλυση της πρώτης δοκιμασίας στο επίπεδο, ο παίκτης καλείται να δημιουργήσει τις βασικές πληροφορίες για τον χαρακτήρα στις δοκιμασίες που ακολουθούν. Μέσα από αυτές τις δοκιμασίες, μαθαίνει για τις μεταβλητές και τους τύπους τους, καθώς και τον ορισμό και ανάθεση τιμής σε αυτές.

Αφού επιλύσει σωστά όλες τις δοκιμασίες του αρχικού επιπέδου, ο παίκτης έχει δημιουργήσει το βασικό σύστημα από όπου μπορεί να λαμβάνει πληροφορίες για τον χαρακτήρα. Το σύστημα εμφανίζει στην οθόνη τις μπάρες ζωής (health), πόντων μαγείας

(mana), εμπειρίας του χαρακτήρα (exp) και επίπεδο ικανοτήτων (level) όπως φαίνεται στην Εικόνα 15. Έπειτα μεταβαίνει στο επίπεδο 2.

5.1.2 Επίπεδο 2

Το περιβάλλον του επιπέδου 2 έχει διαφορετικά γραφικά και κόσμο από το επίπεδο 1 και παρουσιάζεται στην Εικόνα 15.

Εικόνα 15: Περιβάλλον επιπέδου 2

Οι δοκιμασίες και τα σημεία ενδιαφέροντος ακολουθούν την ίδια λογική με το επίπεδο 1. Εξαίρεση, αποτελεί το πρώτο πρόβλημα του επιπέδου όπου ο παίκτης πρέπει να εξοπλιστεί με τον οπλισμό που βρίσκει σε εκείνο το σημείο ώστε να είναι σε θέση να αντιμετωπίσει τους εχθρούς που υπάρχουν στον κόσμο του παιχνιδιού. Η μορφή που έχει στο περιβάλλον του παιχνιδιού παρουσιάζεται πριν και μετά την λύση της δοκιμασίας στις Εικόνες 16 και 17 αντίστοιχα.

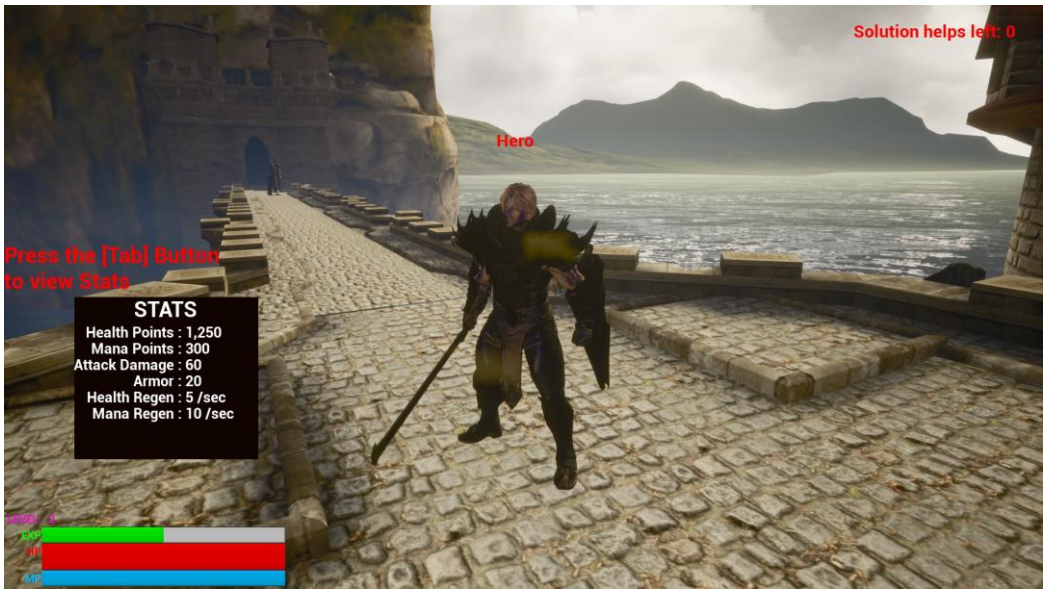
Εικόνα 16: Σημείο ενδιαφέροντος πριν την επίλυση της δοκιμασίας - Επίπεδο 2

Εικόνα 17: Σημείο ενδιαφέροντος μετά την επίλυση της δοκιμασίας - Επίπεδο 2

Στην πρώτη δοκιμασία του επιπέδου ο παίκτης καλείται, έχοντας βρει εξοπλισμό, να αυξήσει τις τιμές των μεταβλητών armor και damage. Με την επίλυση του προβλήματος, ο ήρωας παίρνει στα χέρια του το σπαθί και την ασπίδα που βρήκε και πλέον είναι έτοιμος να αντιμετωπίσει τους εχθρούς του παιχνιδιού.

Στη συνέχεια του ταξιδιού του ήρωα και στην επόμενη δοκιμασία, ο παίκτης με βάση τις μεταβλητές που όρισε στο αρχικό επίπεδο, καλείται να δημιουργήσει μια σελίδα με τα στατιστικά για τον ήρωα. Αυτό γίνεται χρησιμοποιώντας τις κατάλληλες εντολές εκτύπωσης στην Python και συνδυάζοντας μεταβλητές Int και Float με String

ώστε να εμφανίζεται το κατάλληλο περιεχόμενο. Με την επιτυχή έκβαση της δοκιμασίας, ο παίκτης έχει πρόσβαση στη λίστα στατιστικών του χαρακτήρα με το πάτημα του κατάλληλου πλήκτρου.



Εικόνα 18: Αποτέλεσμα Δοκιμασίας 2 - Επίπεδο 2

Αφού ο παίκτης έχει ολοκληρώσει τις δύο πρώτες δοκιμασίες, θα πρέπει να επιλύσει και την τρίτη και τελευταία δοκιμασία του επιπέδου. Εδώ, ο παίκτης μαθαίνει για τις λίστες και συγκεκριμένα την λίστα η οποία θα παίζει τον ρόλο του αποθηκευτικού χώρου (inventory) του χαρακτήρα. Μόλις ο παίκτης καταφέρει να λύσει το παρουσιαζόμενο πρόβλημα προσαρτώντας στην λίστα αντικείμενα, έχει πλέον πρόσβαση στο παράθυρο που του εμφανίζει τα περιεχόμενά της και μπορεί να χρησιμοποιεί. Έπειτα μεταβαίνει στο επόμενο επίπεδο.

Τέλος, αυτό είναι το πρώτο επίπεδο που ο παίκτης έρχεται αντιμέτωπος με εχθρούς. Καθώς έχει λύσει την αρχική δοκιμασία και είναι εξοπλισμένος είναι και σε θέση να τους αντιμετωπίσει, πράγμα αδύνατο στο επίπεδο 1. Πρέπει να σημειωθεί ότι ειδικά για την δοκιμασία 1 του επιπέδου 2, δεν επιτρέπεται στον παίκτη να προχωρήσει αν δεν βρει την επίλυσή του προβλήματος ώστε να εξοπλιστεί. Αυτό αποτελεί θέμα πλαισίου (context) στον κόσμο του παιχνιδιού καθώς σε διαφορετική περίπτωση ο παίκτης θα αντιμετώπιζε εχθρούς αδυνατώντας να τους κερδίσει. Οι εχθροί είναι προγραμματισμένοι να τρέχουν προς τον ήρωα, όταν εισέλθει στο οπτικό τους πεδίο, και να του επιτίθενται. Ο τύπος των εχθρών παραμένει σταθερός σε όλα τα επίπεδα που

ακολουθούν αλλά η δυναμική τους μεγαλώνει βαθμιαία ανάλογα με το επίπεδο που βρίσκεται ο παίκτης. Οι εχθροί του παιχνιδιού, έχουν την μορφή που παρουσιάζεται στις Εικόνες 19 και 20.

Εικόνα 19: Εχθρός τύπου 1

Εικόνα 20: Εχθρός τύπου 2

Ακριβώς πάνω από κάθε εχθρό, βρίσκεται η μπάρα ζωής του, η οποία μειώνεται με τα χτυπήματα και τις ικανότητες του ήρωα και όταν αδειάσει ο εχθρός έχει ηττηθεί και εξαφανίζεται από τον κόσμο.

5.1.3 Επίπεδο 3

Το επίπεδο 3 έχει διαφορετική θεματολογία και περιβάλλον όπως και κάθε επίπεδο του παιχνιδιού αλλά στην αρχή του, αποτελεί προέκταση του επιπέδου 2. Στην Εικόνα 21 παρουσιάζεται το βασικό σκηνικό του επιπέδου 3.

Εικόνα 21: Περιβάλλον επιπέδου 3

Σε αυτό το επίπεδο, όπως και στα επίπεδα 1 και 2, υπάρχουν διάφορα σημεία ενδιαφέροντος που περιέχουν επιπλέον θέματα διδασκαλίας στην Python. Ο παίκτης πρέπει να τα επισκεφτεί και να τα επιλύσει. Παρόμοια με το επίπεδο 2 όμως, υπάρχουν 2 διαφορετικού είδους σημεία ενδιαφέροντος που αποτελούν σεντούκια που πρέπει ο παίκτης να δώσει την σωστή απάντηση στην δοκιμασία και να τα ξεκλειδώσει. Έτσι, μαζί με τις ιδιότητες που προστίθενται στον ήρωα, πρέπει να βρει και τα 3 κομμάτια ενός κλειδιού (key fragments) ώστε στο τέλος του επιπέδου να τα ενώσει και να ξεκλειδώσει την πόρτα για τον επόμενο κόσμο στο ταξίδι. Στις Εικόνες 22 και 23 παρουσιάζονται το πριν και το μετά των δοκιμασιών αυτού του τύπου.

Εικόνα 22: Σημείο ενδιαφέροντος πριν την επίλυση της δοκιμασίας - Επίπεδο 3

Εικόνα 23: Σημείο ενδιαφέροντος μετά την επίλυση της δοκιμασίας - Επίπεδο 3

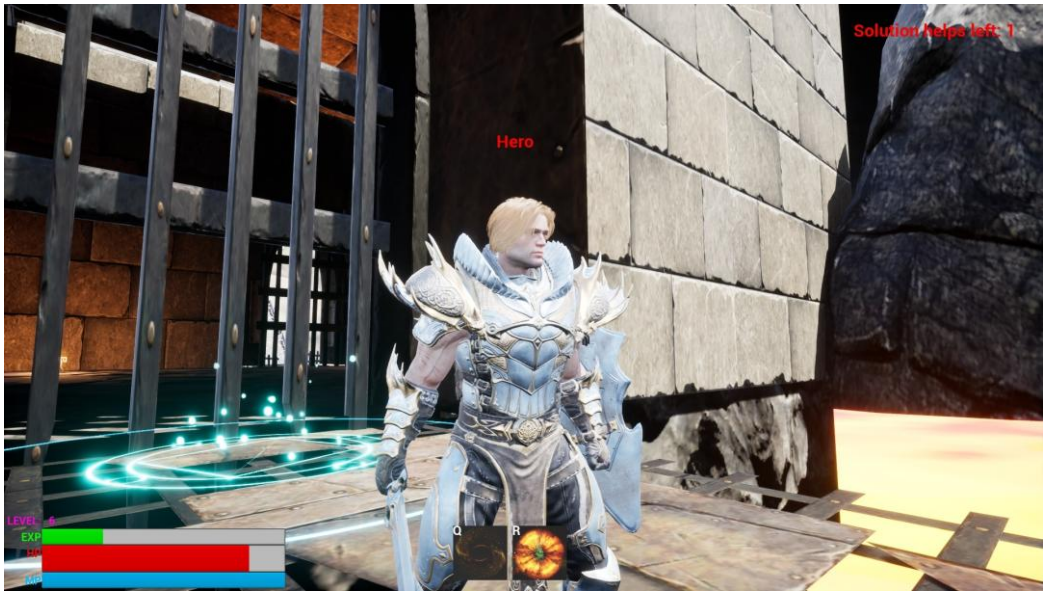
Οι δοκιμασίες που συναντάει ο παίκτης στα σεντούκια σχετίζονται με τη δημιουργία των ικανοτήτων του ήρωα. Αρχικά, στην πρώτη δοκιμασία πρέπει με σωστή κατανόηση και χρήση της δομής if / else σε Python, ο παίκτης να ελέγχει αν έχει αρκετούς πόντους μαγείας (mana) και αν η μεταβλητή του χρόνου επαναφόρτισης της ικανότητας είναι 0 ώστε να χρησιμοποιηθεί. Σε διαφορετική περίπτωση τυπώνεται μήνυμα αδυναμίας εκτέλεσης της ικανότητας. Επιλύοντας τη δοκιμασία, ο ήρωας πλέον έχει πρόσβαση στη δεύτερη ικανότητα και σε περίπτωση που δεν μπορεί να την

χρησιμοποιήσει βάσει των συνθηκών που ελέγχονται στη δοκιμασία, παράγει κατάλληλο ηχητικό μήνυμα. Ταυτόχρονα δίνεται στον παίκτη ένα από τα τρία key fragments.

Αφού ο παίκτης έχει δημιουργήσει για τον ήρωα μια ικανότητα, καλείται στη δοκιμασία που ακολουθεί να δημιουργήσει και την δεύτερη. Αυτή την φορά όμως, γίνεται χρήση της δομής while. Όσο ο ήρωας έχει αρκετό mana και δεν πατήθηκε το κουμπί απενεργοποίησης της ικανότητας, τότε αυτή εκτελείται. Συμπληρώνοντας τον σωστό κώδικα στη δοκιμασία, ο ήρωας έχει πρόσβαση και στην δεύτερη ικανότητά του. Επίσης, ο παίκτης λαμβάνει το δεύτερο key fragment από το σεντούκι.

Όμως, οι νέες προκλήσεις για τον παίκτη δεν σταματούν εδώ. Υπάρχει ένα δωμάτιο του οποίου η πύλη είναι κλειστή και για να εισέλθει σε αυτό ο παίκτης πρέπει να επιλύσει τα προβλήματα εξωτερικά από αυτό. Η τελευταία δοκιμασία που πρέπει να επιλύσει ο παίκτης πριν εισέλθει στο δωμάτιο, σχετίζεται με την κατανόηση της θέσης σε δομές δεδομένων (index). Ο παίκτης καλείται να αναθέσει στη μεταβλητή itemToUse μια τιμή από την λίστα inventory με βάση το index της.

Λύνοντας και τη τελευταία αυτή εξωτερική δοκιμασία, η πύλη ανοίγει αυτόματα και ο ήρωας κάνει χρήση ενός φίλτρου από το inventory του. Μόλις όμως, ο παίκτης εισέλθει στο δωμάτιο όλες οι πύλες κλείνουν. Εκεί ο ήρωας βρίσκεται αντιμέτωπος με εχθρούς τους οποίους ο παίκτης δεν μπορεί να έχει δει και προβλέψει. Για τον λόγο αυτό η τελευταία δοκιμασία, πριν ανοίξουν οι πύλες, προετοιμάζει τον παίκτη με το εκπαιδευτικό περιεχόμενο να τον οδηγεί στην χρήση κάποιου αντικειμένου. Στις Εικόνες 24 και 25 παρουσιάζεται η εξωτερική πλευρά της τοποθεσίας και η εσωτερική αντίστοιχα.



Εικόνα 24: Δοκιμασία ανοίγματος πύλης - Επίπεδο 3

Εικόνα 25: Εσωτερικά του δωματίου - Επίπεδο 3

Αφού ο παίκτης νικήσει όλους τους εχθρούς στο δωμάτιο, οι πύλες ανοίγουν και του δίνεται το τρίτο και τελευταίο key fragment. Στην τελευταία δοκιμασία του επιπέδου, πρέπει με χρήση υποσυνόλου της λίστας inventory, ο παίκτης να ανακτήσει τα τρία key fragments που βρήκε ώστε να μπορέσει να ξεκλειδώσει την είσοδο για το επόμενο επίπεδο. Μόλις λυθεί και αυτή η δοκιμασία, ο ήρωας μεταφέρεται στο επόμενο επίπεδο.

5.1.4 Επίπεδο 4

Σε όλα τα προηγούμενα επίπεδα, η δυσκολία που συναντούσε κάποιος παίκτης σχετιζόταν μόνο με την αντιμετώπιση των εχθρών αλλά και την επίλυση των δοκιμασιών. Στο επίπεδο 4, εισάγεται και ένα νέο είδος δυσκολίας που αφορά τον κόσμο του παιχνιδιού. Εάν ο παίκτης δεν είναι προσεκτικός μπορεί να πέσει εκτός πλατφόρμας-κόσμου και να οδηγηθεί σε ήττα άμεσα με την εμφάνιση σχετικού μηνύματος “Game Over”. Το ίδιο μήνυμα θα εμφανιζόταν και στον παίκτη στην περίπτωση που δεχτεί μεγάλο ποσοστό ζημιάς (damage) από τους εχθρούς. Στην Εικόνα 26 παρουσιάζεται το περιβάλλον του επιπέδου 4 και στην Εικόνα 27 το σχετικό μήνυμα ήττας προς τον παίκτη.

Εικόνα 26: Περιβάλλον επιπέδου 4

Τα σημεία ενδιαφέροντος και ο τρόπος λειτουργίας τους στο συγκεκριμένο επίπεδο, δεν διαφέρουν από ότι έχει συναντήσει ο παίκτης στο ταξίδι του στο παιχνίδι.

Εικόνα 27: Μήνυμα ήττας "Game Over" προς τον παίκτη

Με την εμφάνιση του συγκεκριμένου μηνύματος στην οθόνη, δίνονται 3 επιλογές στον παίκτη. Με την πρώτη επιλογή "Restart from Last Save" ο παίκτης θα επανέλθει στο σημείο που αποθηκεύτηκε το παιχνίδι τελευταία φορά και συγκεκριμένα για το επίπεδο 4 αυτό είναι η αρχή του επιπέδου. Με την δεύτερη επιλογή "Return to Main Menu", ο παίκτης μπορεί να επιστρέψει στην αρχική οθόνη μενού που συναντάει κάθε φορά κατά την εκκίνηση του παιχνιδιού και με την τρίτη επιλογή "Quit" ο παίκτης μπορεί να αποχωρήσει εντελώς από την εφαρμογή, τερματίζοντας την λειτουργίας της.

Σε αυτό το επίπεδο δεν υπάρχουν ιδιαιτερότητες στα σημεία ενδιαφέροντος και στον τρόπο λειτουργίας του παιχνιδιού αλλά χρειάζεται προσοχή από πλευράς παίκτη να μην φύγει έξω από τα όρια και βρεθεί σε πτώση που θα τον οδηγήσει στην ήττα. Ο ήρωας πρέπει να επισκεφτεί όλα τα checkpoint και να λύσει τις δοκιμασίες ενώ παράλληλα πρέπει να μάχεται τους εχθρούς που βρίσκονται στο επίπεδο και προσπαθούν να τον εμποδίσουν.

Αρχικά ο ήρωας, έχει συγκεντρώσει κάποια αντικείμενα στην λίστα inventory, τα οποία αποτελούν απόδειξη της προόδου του παίκτη στο ταξίδι του ήρωα σε φανταστικούς κόσμους. Έτσι, σε αυτή τη δοκιμασία ο παίκτης χρησιμοποιώντας δομή for loop σε Python, καλείται να αυξήσει την τιμή της μεταβλητής εμπειρίας (exp) του ήρωα κατά 1 για κάθε αντικείμενο στη λίστα inventory που δεν είναι φίλτρο (potion). Εκτός, από την δομή for, ο παίκτης διδάσκεται και για την επανάληψη (iteration) με βάσει κάποια ακολουθία είτε είναι λίστες είτε Strings.

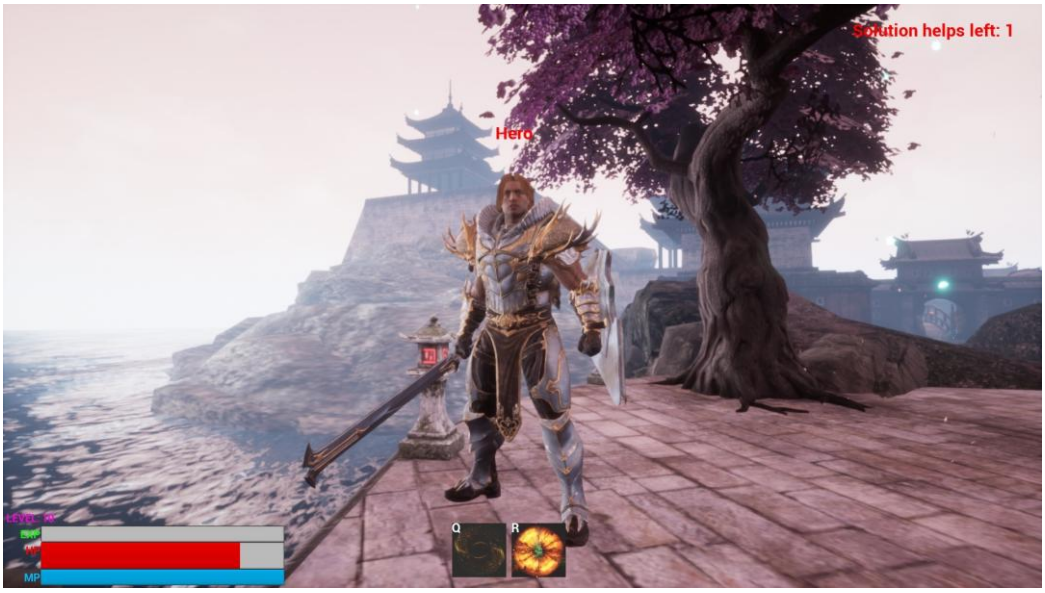
Στη δεύτερη δοκιμασία που συναντάει στο επίπεδο ο παίκτης, μαθαίνει για τη δομή δεδομένων dictionary στην Python, και καλείται να επανασχεδιάσει τη λειτουργία της σελίδας με στατιστικά που έφτιαξε στην δοκιμασία 2 του επιπέδου 2. Ο παίκτης μαθαίνει για τις ιδιότητες των Dictionaries και βασιζόμενος σε αυτά που έμαθε σε προηγούμενα επίπεδα αλλά και στην αρχική δοκιμασία του επιπέδου 4, αρχικοποιεί το dictionary με τους κατάλληλους συνδυασμούς κλειδιών : τιμών (key : value). Επίσης, πρέπει να ανακτήσει τις τιμές και τα κλειδιά σε όλη την ακολουθία που δημιούργησε με δομή for. Έπειτα τα τυπώνει με τον ίδιο τρόπο που διδάχθηκε στο επίπεδο 2.

Στην τελευταία δοκιμασία του επιπέδου, ο παίκτης διδάσκεται για τη λειτουργία και τον τρόπο δημιουργίας των συναρτήσεων σε Python. Με βάση τις ικανότητες που δημιούργησε στο επίπεδο 3, γίνεται προσπάθεια επεξήγησης της δημιουργίας και χρήσης ενός function. Στον παίκτη παρουσιάζονται τα προτερήματα μιας τέτοιας επιλογής όπως η επαναχρησιμοποίηση λειτουργιών και κώδικα και καλείται να δημιουργήσει ένα function που θα χρησιμοποιείται για την πρώτη ικανότητα του ήρωα.

Με την επιτυχή υλοποίηση του function που απαιτείται από τον παίκτη, ο ήρωας μεταφέρεται στο επόμενο επίπεδο.

5.1.5 Επίπεδο 5

Το επίπεδο 5 είναι το τελευταίο επίπεδο του παιχνιδιού και περιέχει τις δυσκολότερες δοκιμασίες καθώς πραγματεύεται πιο πολύπλοκα θέματα της γλώσσας προγραμματισμού Python.



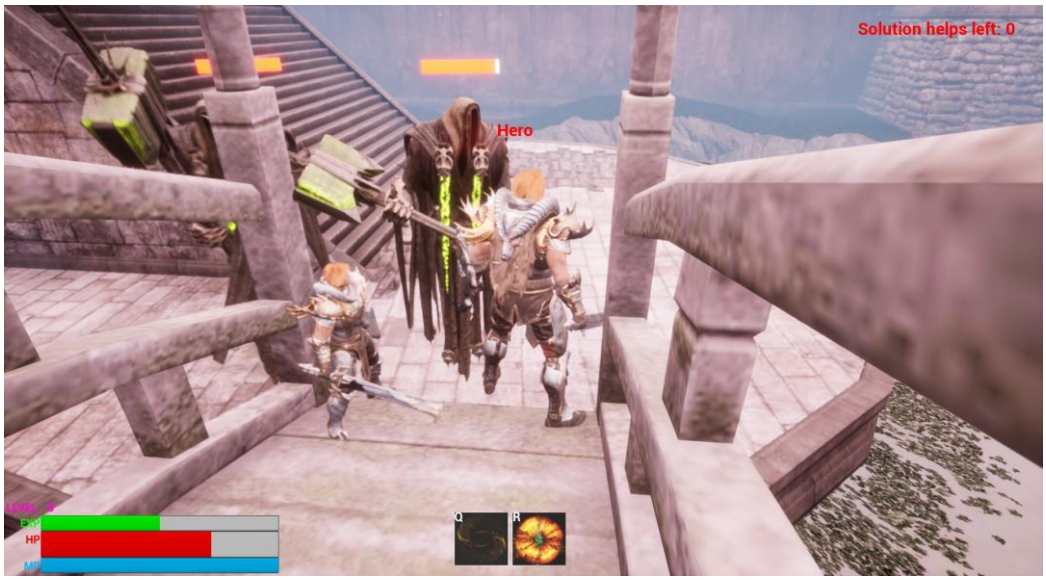
Εικόνα 28: Περιβάλλον επιπέδου 5

Ο παίκτης δεν συναντάει κάποια ιδιαίτερη λειτουργία στο σύστημα του παιχνιδιού που δεν έχει ξαναδεί σε προηγούμενα επίπεδα. Οι εχθροί σε αυτό το επίπεδο έχουν την μέγιστη δυναμική τους και ο παίκτης αν δεν είναι προσεκτικός μπορεί να παγιδευτεί στο περιβάλλον του επιπέδου από τους εχθρούς χωρίς διέξοδο και με κίνδυνο την ήττα.

Στην πρώτη δοκιμασία που συναντάει στο επίπεδο, ο παίκτης διδάσκεται για την έννοια της αντικειμενοστρέφειας και τον τρόπο χρήσης της στη γλώσσα προγραμματισμού Python που είναι μια αντικειμενοστρεφής γλώσσα προγραμματισμού. Στη συνέχεια παρέχονται κατευθύνσεις και παραδείγματα και ο παίκτης καλείται πλέον να δημιουργήσει μια κλάση για τα φίλτρα (potions) που είχε αποθηκεύσει στη λίστα inventory ως Strings μέχρι εκείνο το σημείο του παιχνιδιού. Με αυτό τον τρόπο γίνεται προσπάθεια ο παίκτης να κατανοήσει τη δύσκολη αυτή έννοια προγραμματισμού, ιδιαίτερα για άτομα χωρίς κάποιο προγραμματιστικό υπόβαθρο.

Έχοντας εισάγει τον παίκτη στις έννοιες της αντικειμενοστρέφειας, ακολουθεί δοκιμασία που έχει ως εκπαιδευτικό περιεχόμενο την κληρονομικότητα. Με τη θεωρία που παρέχεται στον παίκτη και τα παραδείγματα που περιέχει, γίνεται προσπάθεια εμπέδωσης τον εννοιών που παρουσιάστηκαν στην δοκιμασία 1 του επίπεδου και επέκτασής τους. Σε αυτό το σημείο, στον παίκτη δίνεται ένα παράδειγμα κλάσης για τον κεντρικό ήρωα και ζητείται η δημιουργία μιας κλάσης (HelperHero) που θα κληρονομεί

από αυτή του ήρωα. Στόχος είναι με την κατανόηση της θεωρίας και των παραδειγμάτων που δίνονται ο παίκτης να εισάγει τον σωστό κώδικα ώστε η υποκλάση να έχει την συμπεριφορά που αναμένεται. Με την επιτυχή επίλυση της δοκιμασίας εμφανίζεται στον κόσμο ένα μικρότερος ήρωας (HelperHero) που τον βοηθάει για ένα μικρό χρονικό διάστημα στην μάχη του με τους εχθρούς.



Εικόνα 29: Αποτέλεσμα επίλυσης δοκιμασίας 2 και εμφάνιση "HelperHero"

Έπειτα ο παίκτης, ανεβαίνοντας τα σκαλοπάτια προς τον τερματισμό, συναντάει ένα σημείο ενδιαφέροντος που περιέχει ένα σπαθί. Μόλις ενεργοποιηθεί η λειτουργία της δοκιμασίας, μαθαίνει ότι δεν μπορεί να πάρει και να χρησιμοποιήσει αυτό το “θρυλικό” ξίφος καθώς στην κλάση ήρωα που υπάρχει και συγκεκριμένα στο dictionary με τα όπλα και τα επίπεδα ζημιάς τους, δεν έχει οριστεί κλειδί (key) “legendary” που να αντιστοιχεί σε μια τιμή (value) επιπέδου ζημιάς (damage). Σε περίπτωση που προσπαθούσε να το πάρει, η Python θα επέστρεφε σφάλμα (Error) και του παρέχεται η θεωρία και τα παραδείγματα για την διαχείριση τέτοιων καταστάσεων με χρήση δομών try / except σε Python. Στη δοκιμασία ο παίκτης καλείται να γράψει τον κώδικα που θα διαχειριστεί την κατάσταση ανύπαρκτης τιμής κλειδιού στο dictionary με τα είδη σπανιότητας των όπλων και τις τιμές ζημιάς τους, εισάγοντας την κατάλληλη τιμή κλειδιού για το όπλο που μόλις βρήκε συνδυάζοντας τις γνώσεις που έλαβε από τη δοκιμασία ανάλυσης και χρήσης dictionaries.

Εικόνα 30: Σημείο ενδιαφέροντος "θρυλικού" σπαθιού

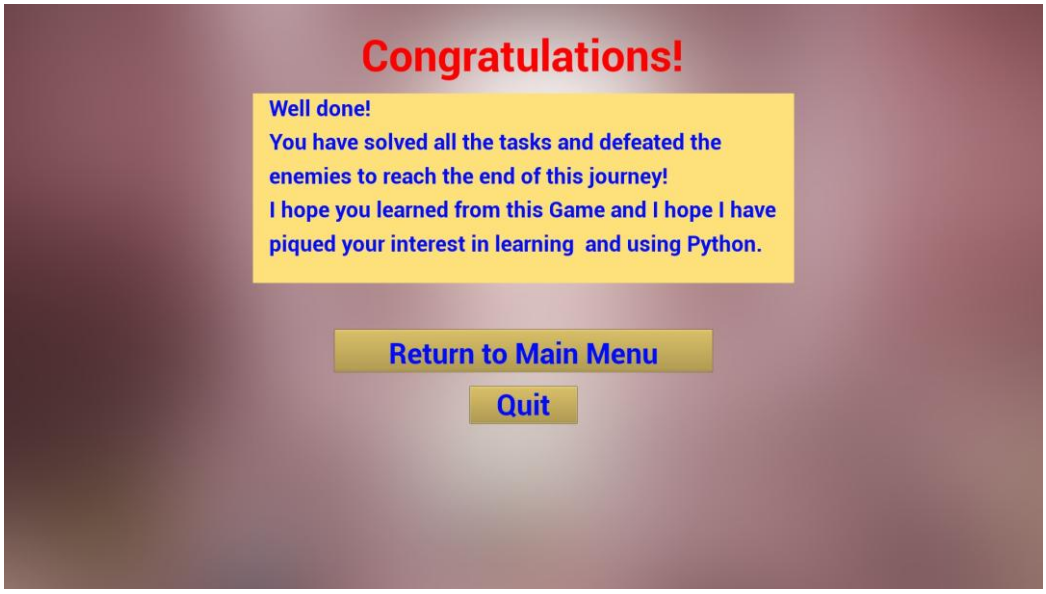
Με τη σωστή επίλυση της δοκιμασίας ο παίκτης μπορεί να συνεχίσει κρατώντας το “θρυλικό” σπαθί με τα μεγάλα επίπεδα ζημιάς που μόλις βρήκε.

Εικόνα 31: Αποτέλεσμα επίλυσης δοκιμασίας 3

Συνεχίζοντας το ταξίδι του ήρωα, ο παίκτης φτάνει στο υψηλότερο σημείο που τον οδηγούν τα σκαλοπάτια και την τελευταία δοκιμασία του επιπέδου αλλά και του παιχνιδιού. Σε αυτό το σημείο μαθαίνει για τα πακέτα και modules που υπάρχουν στην Python και τις δυνατότητες που τις επιτρέπουν να έχει. Ταυτόχρονα, καθώς θα ήταν

αδύνατο να καλυφθούν όλα τα πακέτα και modules της Python, στα πλαίσια ενός παιχνιδιού σοβαρού σκοπού, γίνεται παρότρυνση εξερεύνησης των πακέτων που δεν συμπεριλαμβάνονται στη γλώσσα προεγκατεστημένα (built-ins) και επεκτείνουν την λειτουργικότητα και χρησιμότητα της Python σε μεγάλο βαθμό.

Στην τελική αυτή δοκιμασία, ο παίκτης μαθαίνει επίσης για τη δημιουργία από τον ίδιο, αν είναι αναγκαίο, πακέτων και modules και τις δυνατότητες επαναχρησιμοποίησής τους σε άλλα έργα (project). Ο παίκτης καλείται να εισάγει στον κώδικα ένα module που έχει ήδη υλοποιηθεί και υπάρχει στην βιβλιοθήκη της Python του παιχνιδιού. Με την σωστή εισαγωγή και χρήση του module, γίνεται έλεγχος από τον κώδικα αν αυτό είναι το τελευταίο επίπεδο και ως αποτέλεσμα επιστροφής True, ο τερματισμός του παιχνιδιού.



Εικόνα 32: Μήνυμα τέλους του παιχνιδιού

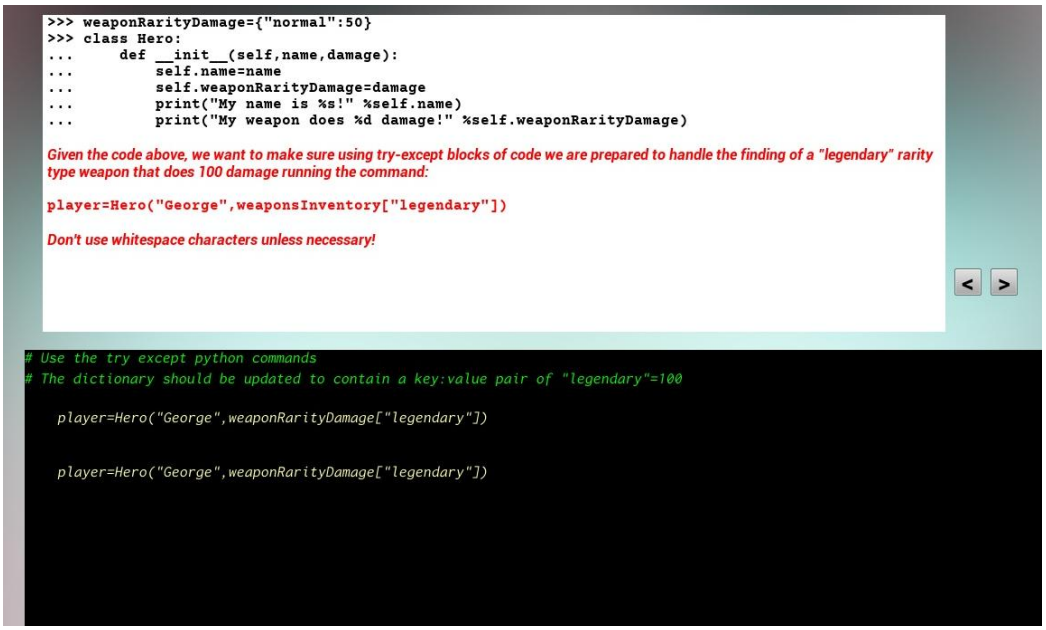
Στο μήνυμα που εμφανίζεται κατά τον τερματισμό του παιχνιδιού, δίνονται συγχαρητήρια στον παίκτη και του παρέχονται οι επιλογές επιστροφής στο κύριο μενού με το κουμπί “Return to Main Menu” και εξόδου και τερματισμού της λειτουργίας της εφαρμογής με το κουμπί “Quit”.

5.2 Περιεχόμενα παιχνιδιού

Σε κάθε επίπεδο παρουσιάζεται στον παίκτη θεωρία και παραδείγματα για το αντικείμενο που πραγματεύεται η δοκιμασία. Έπειτα, ακολουθεί με κόκκινα γράμματα η

εκφώνηση του προβλήματος που καλείται ο παίκτης να λύσει και ορίζεται ο τρόπος με τον οποίο θα το κάνει. Τέλος, οι παίκτες παροτρύνονται να μην χρησιμοποιούν κενά στον κώδικα ενώ έχουν αυτή την δυνατότητα καθώς παρόλο που ο κώδικας θα είναι συντακτικά σωστός σε Python, η χρήση πολλών ανούσιων κενών οδηγεί σε μείωση της αναγνωσιμότητας του κώδικα.

Στο πεδίο τύπου κονσόλας που μπορεί να εισάγει κώδικα ο παίκτης, δίνονται περαιτέρω διευκρινίσεις και κατευθύνσεις στον παίκτη με την μορφή σχολίων στον κώδικα και σε κάποιες περιπτώσεις υπάρχει μέρος του κώδικα το οποίο πρέπει ο παίκτης να συμπληρώσει ώστε να επιλύσει τη δοκιμασία.



Εικόνα 33: Θεωρία - εκφώνηση προβλήματος - βοηθητικά σχόλια

Η θεωρία, οι ορισμοί των προβλημάτων καθώς και τα σχόλια με τον συμπληρωμένο κώδικα, αν υπάρχει, αλλά και μία ενδεικτική λύση παρουσιάζονται στα επόμενα κεφάλαια για κάθε επίπεδο και δοκιμασία επιπέδου του παιχνιδιού.

5.2.1 Περιεχόμενα επιπέδου 1

Πίνακας 3: Εκπαιδευτικό περιεχόμενο επιπέδου 1

	A/A	Επεξήγηση του εκπαιδευτικού περιεχομένου στο παιχνίδι και παραδείγματα	Εκφώνηση προβλήματος

Δοκιμασία	**1**	Like another programming language, Python can print something that we want on the screen.
		In order to do that we call the function print( ) and inside the parentheses, inside double quotation marks we can write what we want to be printed.
		Even though we can use single quotation marks, using double can allow us to print a single quotation mark that would otherwise cause errors.
		>>> print("It's nice to see you")
		It's nice to see you
		>>> print('It's nice to see you')
		error
		Therefore from now on, we will be using double quotation marks when we want to print or initialize a String.
			Here we want to print the name for our Hero, so print how you feel your Hero should be called.

	2	In Python we can store values in variables. One type of numeric data are Integers (whole numbers without decimals). Variables are defined the moment we assign a value to them: >>> aVar = 5 This means that the variable [aVar] has the value of [5]. In Python using the symbol [#] means that everything in the same line after that is considered a comment so Python will ignore it and not execute it. Sometimes it is important to comment for future reference but correct and valid code won't be executed if it is commented!	In this task we are trying to set our initial health points [health] to the value of 1000.

	3	We can define and set multiple variables in the same line like: >>> x, y = 2, 3	Same as before we want to set the variables of our character's [mana] and [damage]. Try setting those in the same line to the values of [200] and [50] respectively.
	4	Another type of numeric data is a floating point number (float). These are numbers with decimals. Float type variables can be initialized like: >>> var = 9.2 This means that the variable [var] has the value of [9.2].	Here we are trying to set the variables of [exp] and [armor] to [7.5] and [10] respectively. Try setting those in the same line as we did before.

Κώδικας 1: Κώδικας - σχόλια δοκιμασίας 1 - επίπεδο 1

```
# You can have nothing between the quotation marks or even call print() without a
ny arguments and it would be valid for Python
# In that case our hero would sadly have no name
```

Κώδικας 2: Ενδεικτική λύση δοκιμασίας 1 - επίπεδο 1

```
print("Hero")
```

Κώδικας 3: Κώδικας - σχόλια δοκιμασίας 2 - επίπεδο 1

```
# Set the variable health to 1000
# Afterwards print it on the screen with the print() function like: print(aVar)
# We don't need quotation marks when printing variables
```

Κώδικας 4: Ενδεικτική λύση δοκιμασίας 2 - επίπεδο 1

```
health = 1000
print(health)
```

Κώδικας 5: Κώδικας - σχόλια δοκιμασίας 3 - επίπεδο 1

```
# x,y=2,3
# Set the variable mana to 200 and damage to 50
```

Κώδικας 6: Ενδεικτική λύση δοκιμασίας 3 - επίπεδο 1

```
mana, damage = 200, 50
```

Κώδικας 7: Κώδικας - σχόλια δοκιμασίας 4 - επίπεδο 1

```
# a,b = 1,3.4
# Set the variable exp to 7.5 and the variable armor to 10
```

Κώδικας 8: Ενδεικτική λύση δοκιμασίας 4 - επίπεδο 1

```
exp, armor = 7.5, 10
```

5.2.2 Περιεχόμενα επιπέδου 2

Πίνακας 4: Εκπαιδευτικό περιεχόμενο επιπέδου 2

	Α/Α	**Επεξήγηση του εκπαιδευτικού περιεχομένου στο παιχνίδι και παραδείγματα**	**Εκφώνηση προβλήματος**
Δοκιμασία	**1**	In Python we can add, subtract, multiply and divide with the symbols: + , - , * , / We can also get exponential results with: ** and the modulus (the remainder of an Integer division) with: %. In case we want to add or subtract or multiply or divide the value of a variable with another number and the original variable would become the result, we can use: += , -= , *= , /= respectively. For example if x=2 and we want to add 3 to it we can write: x+=3 which would make x equal to 5.	We have just found a shield and a sword for our Hero. This calls for a boost to our armor and damage. Try adding to the [armor] and [damage] variables the number 10.

	2	In Python we can combine String variables. For example, the following block of code:
		>>> a = "health"
		>>> b = "points"
		>>> print(a + " " + b)
		health points
		But we can't combine Strings with Ints or Floats. (Ints with Floats would work but the result would always be a float). We can print a String that contains the value of a float variable with the following code:
			Here we are trying to format Strings with numbers to create a stats page for our character.
			We want to view "Health Points", "Mana Points", "Attack Damage", "Armor", "Health Regen" and

		>>> afloatvar = 100.0
		>>> print("health points: %f" %afloatvar)
		which will print: health points: 100.0
		We can use the symbols %f , %d, %s like we did in the previous example, to print as Floats, Integers and Strings respectively.
			"Mana Regen". We also want health to be printed as a float, regardless of the original value type and the rest to be printed as Ints, so make sure to use the right symbols. Each print command needs to be in its own line!

	3	In order to store multiple items in memory we can use Python lists.
		Python lists can store any variable and multiple types of variables can be stored in the same list.
		Here, we are trying to create a list that would be our inventory and would store our potions.
		For now let's just name them as: [healthPot], [manaPot]
		and store them as Strings indicating the different use each one will have.
		Lists can be initialized like:
		>>> inventory = [ ]
		or even initialized to contain something like:
		>>> inventory = ["a"]
		We can append items to the end of the list using the listName.append(item) method.
			Append the item [manaPot] to the list inventory.

Κώδικας 9: Κώδικας - σχόλια δοκιμασίας 1 - επίπεδο 2

```
# x = 1
# x = x+2 or x+=2 results in x becoming 3.

# Make the armor addition below this:

# Make the damage addition below this:
```

Κώδικας 10: Ενδεικτική λύση δοκιμασίας 1 - επίπεδο 2

```
# x = 1
# x = x+2 or x+=2 results in x becoming 3.

# Make the armor addition below this:
armor += 10
# Make the damage addition below this:
damage += 10
```

Κώδικας 11: Κώδικας - σχόλια δοκιμασίας 2 - επίπεδο 2

```
# The variables names are: health, mana, attackDamage, armor, hpregen, mpr
egen
# Print the exact Strings with the corresponding variable in the exact ord
er they are given
# Printed strings should have the format: "String: value"
# Note that hpregen and mpregen happen per second therefore combining is n
eeded to print like: "String: value/sec"
print("Health Points: %f" %health)
```

Κώδικας 12: Ενδεικτική λύση δοκιμασίας 2 - επίπεδο 2

```
print("Health Points: %f" %health)
print("Mana Points: %d" %mana)
print("Attack Damage: %d" %attackDamage)
print("Armor: %d" %armor)
print("Health Regen: %d/sec" %hpregen)
print("Mana Regen: %d/sec" %mpregen)
```

Κώδικας 13: Κώδικας - σχόλια δοκιμασίας 3- επίπεδο 2

```
# alist.append("potion")
# Append the item manaPot as String to the list inventory
inventory=["healthPot"]
```

Κώδικας 14: Ενδεικτική λύση δοκιμασίας 3 - επίπεδο 2

```
inventory=["healthPot"]
inventory.append("manaPot")
```

5.2.3 Περιεχόμενα επιπέδου 3

Πίνακας 5: Εκπαιδευτικό περιεχόμενο επιπέδου 3

	A/A	Επεξήγηση του εκπαιδευτικού περιεχομένου στο παιχνίδι και παραδείγματα	Εκφώνηση προβλήματος
Δοκιμασία	**1**	In Python we can use conditions to determine the validity of a statement. A condition can be either True or False. The symbols used are : = = for equality != not equal < less than > greater than < = less than or equal >= greater than or equal Python also has operators to combine conditions created with the symbols above. Those operators are: and or These operators are case sensitive! Combining conditions with operators can have one of the following results: True and True results in True True and False results in False False and False results in False True or True results in True True or False results in True False or False results in False In order to create code that will execute if some conditions are met, we use the if statement. The following code block: >>> x,y = 2,1	Knowing all that, here we want to check for certain conditions to see if we can use an ability. Those conditions are if we have enough mana (70 needed at least) and if the ability's cooldown is 0.

		>>> if x>y:
		... print("Conditions met")
		would print: Conditions met
		In this case the print command will happen only if the condition is met. This code or any other code inside an if/else (loops, functions, classes etc) is indicated by indentation. We can add indentation by pressing [Tab] or 4 Spaces [Spacebar] but the two can't be mixed. In Python 3 the use of spaces is preferred and in this game [Tab] has been configured to add 4 spaces.
		Also we might want to check for another condition in case the first one isn't met with elif:
		>>> if x>y:
		... print("Conditions met")
		>>> elif x<y:
		... print("Secondary conditions met")
		In case no conditions are met we might need to do something anyway and this is done with else:
		>>> if x>y:
		... print("Conditions met")
		>>> elif x<y:
		... print("Secondary conditions met")
		>>> else:
		... print("No conditions met")
		And if needed we could add more if/else clauses inside other if/else by adding one more step of indentation.	

	2	In Python we also have the ability to create a loop that would run for a specified amount of times (or even time).
		This can be done using a While loop like:
		>>> a = 0
		>>> while a<5:
		>>> print(a)
			Here we are checking, if we have enough mana. The ability is stopped if we don't have enough mana or if the player presses the ability's button to

		>>> a = a + 1
		This would print:
		0
		1
		2
		3
		4
		Using the previous loop logic, another way to stop the loop from running is using the break command like:
		>>> while True:
		... print(a)
		... a = a + 1
		... if a>4:
		... break
		The loop above would have the same result as the previous one. Another thing to note is that the statement while True is always True and since there is no variable changing on the while condition without the break command the loop would run forever!
		On the other hand the statement while False would never execute the loop.
		Like we did with the last ability, we want to check for conditions and if those conditions are met, then use our ability spell.
		The difference this time is that we won't be doing it once if the conditions are met but instead keep executing the function while those conditions are True. In other words keep the ability running while the conditions are True.
			manually stop it.
			This ability though doesn't come for free, it uses some mana and we should be subtracting the ability's mana cost each loop until we don't have enough mana or the players stops it.
			The mana cost here is 10.
			Be careful with the indentation!

	3	In Python lists (or other data storing structures) items can be retrieved by their index or in other words the position they have in the list. Python like many other programming languages uses a	Here, to prepare for what's to come, we want to assign to the variable itemToUse our

		zero based indexing system. This means that the position of the first item is 0, the second's is 1 and the third's is 2 and so on. Given a list, the python command to print() the third item would be:
		>>> someList = ["a","b","c"]
		>>> print(someList[2])
		c
			manaPot from the list inventory in order to restore our mana Points.

	4	We know that python uses a 0-based indexing system. Python can also use negative-indexing system. For example given aList=[1,2,3,4,5], we know that aList[0] corresponds to the number 1 that is indexed first. With negative-indexing, every element in the list gets a negative index starting from index -1 which corresponds to the last element. The second from the end element has an index of -2, the third from the end has -3 and so on. This is very useful in case we need a specific element from the end but we don't actually know the size of the list and therefore the index of the last element. In those cases we simply use: aList[-1] if we want to get the last element. Lists and other data storing structures are not the only ones Python has indexes for. Strings also have indexes. For example the String: name="George" can be accessed using indexes. If we use the command: >>> print(name[0]) it will print: G In the command above, name[0] corresponds to the single-letter String that is first and in this case it's "G". If we had to use negative-indexing we can say that name[0] is equal to name[-6]. In some cases, we might need to get a subset of a sequence of elements, for example in a list.	Here we need to print the items starting from the third up to how many they might be in our inventory, excluding the last one, since we were given a potion, in order to get our key fragments.

		Let's assume that by mistake or because we didn't know where else to store some elements, we have appended to our list named inventory some random unrelated items. At this point we don't know how many random items got in there but we know that each got appended and we also haven't found any more potions.
		Using this information we know that the first two elements in the list are our potions and we could use:
		>>> print(inventory[0])
		it will print: "healthPot"
		>>> print(inventory[1])
		it will print: "manaPot"
		This might be convenient here but what if we had potion items in the first 10000 indexes of the list? And what if we had an equally big number of random items after them?
		This is where we can use slicing to get a subset of the original list. We do this by creating a command like: aList[1:3].
		This means that we will access elements from index 1 (inclusive) up to index 3 (exclusive). The following code:
		>>> aList=[1,2,3,4,5]
		>>> anotherList=aList[1:3]
		>>> print(anotherList)
		it will print: [2,3]
		This can work with negative indexes too:
		>>> anotherList=aList[2:-1]
		>>> print(anotherList)
		will result in printing: [3,4]
		And if we needed all the elements from a specific index to the end of the list we could do:
		>>>anotherList=aList[2:]	

		>>> print(anotherList) and the result would be: [3,4,5]	

Κώδικας 15: Κώδικας - σχόλια δοκιμασίας 1- επίπεδο 3

```
# Create the conditions and use the necessary operators to check if they are met
# If the conditions are met we execute: ultimateAbility()
# Otherwise we print "Impossible action"
# The variables are mana and abilityCooldown

    ultimateAbility()
    mana = mana - 70

    print("Impossible action")
```

Κώδικας 16: Ενδεικτική λύση δοκιμασίας 1 - επίπεδο 3

```
if mana >= 70 and abilityCooldown == 0:
    ultimateAbility()
    mana = mana - 70
else:
    print("Impossible action")
```

Κώδικας 17: Κώδικας - σχόλια δοκιμασίας 2- επίπεδο 3

```
# Variable mana has our current mana points

    ability()

    if buttonPressed():
    break
```

Κώδικας 18: Ενδεικτική λύση δοκιμασίας 2 - επίπεδο 3

```
while mana>=10:
    ability()
    mana-=10
    if buttonPressed():
    break
```

Κώδικας 19: Κώδικας - σχόλια δοκιμασίας 3- επίπεδο 3

```
# inventory = ["healthPot","manaPot"]
# Assign to the variable itemToUse the manaPot from our inventory list
```

Κώδικας 20: Ενδεικτική λύση δοκιμασίας 3 - επίπεδο 3

```
itemToUse=inventory[1]
```

Κώδικας 21: Κώδικας - σχόλια δοκιμασίας 4- επίπεδο 3

```
# List named inventory has already been created
```

Κώδικας 22: Ενδεικτική λύση δοκιμασίας 4 - επίπεδο 3

```
print(inventory[2:-1])
```

5.2.4 Περιεχόμενα επιπέδου 4

Πίνακας 6: Εκπαιδευτικό περιεχόμενο επιπέδου 4

	A/A	**Επεξήγηση του εκπαιδευτικού περιεχομένου στο παιχνίδι και παραδείγματα**	**Εκφώνηση προβλήματος**
Δοκιμασία	**1**	In python we might have the need to see or check every item in a sequence. To do that we can use a for loop. For loops can iterate for a number of times or over a sequence that might be a list (or other data storing structures) or even a String. We can do this like following: >>> for a in range(3): ... print(a) The code above will print the numbers: 0 1 2 Number 3 is not printed because python uses zero-based indexing which means indexing starts from 0 and the first 3 numbers the variable will take are 0, 1, 2 and they get printed. We also have the ability to stop a for loop or a while loop whenever we want. We can do that by using break. If we need to stop the execution of a loop when some condition(s) are met then we can: >>>for i in range(5): ... print(i) ... if i = = 3: ... break	Here we want to loop over every item in our inventory list that is not a potion and as a reward we will be adding 1 to our exp for each item found. The list named inventory is given and at this point we might not know what or how many items it contains. This doesn't make a difference for Python as long as the list is defined before the loop. Don't worry about the break statement here, it's there just in case someone loops for an infinite iterable.

		The above code would result in: 0 1 2 3 Here we started executing the loop like in the previous example but when the variable [i] became equal to 3 we executed break that was nested inside the if statement. If we didn't nest our break inside the if statement the loop would execute only once since it would print(i) and then find the break command that stops the loop. To iterate over a String we do the following: >>> for x in "game": ... print(x) The code above when executed will print: g a m e In this case the variable x took the value of each letter in the String given.	

	2	Another data structure to store data in Python are dictionaries.
		Dictionaries store date in pairs: {key:value}. Each key corresponds to one value and we can retrieve the value using that key.
		Dictionaries contents are not ordered like lists and can't take duplicates. If we insert a duplicate key: that corresponds to some value, it would replace the value of the old key.
		Dictionaries can be initialized like:
		>>> age={ }
		>>> age["George"] = 25}
			We will be setting the values for each key from the variables: health, mana, attackDamage, armor, hpregen, mpregen
			The names of variables are also case sensitive!
			Since dictionaries can't be ordered afterwards, make sure you put the keys:values in the order

		>>> age["Nick"] = 30}
		>>> age["Bob"] = 35}
		or :
		>>> age={"George":25, "Nick":30, "Bob":35}}
		>>> print(age)}
		{'George': 25, 'Nick': 30, 'Bob': 35}
		Or if we want to know the age of a specific person by their name (key):
		>>> print(age["George"])
		25
		Key values are case sensitive!
		In order to iterate over all dictionary key:values we can do the following:
		>>> x=0
		>>> for name,years in age.items( ):
		>>> x=x+years
		>>> averageAge=x/len(age)
		30.0
		len( ) is a function that can be used to return the number of {key:value} pairs in the given dictionary (or other data structures) and is built-in Python.
		In a previous task we created a stats page for our character by printing strings and values.
		Since the strings don't change and for example when we say health we always refer to our character's health points it would be more appropriate to create a dictionary containing these values.
		The dictionary created should use the keys: "Health Points", "Mana Points", "Attack Damage", "Armor", "Health Regen" and "Mana Regen"
		Keys are case sensitive!
			given since from Python version 3.7 dictionaries are maintained and guaranteed insertion order.
			After creating the dictionary, iterate over its keys:values and print each pair like we did when creating the stats info page.

	3	Another very useful thing we can do in Python is creating functions.	The fact that we managed to cast our

		Functions are blocks of code which can be executed by calling the function name. This makes our code reusable and easier to read and understand.
		In previous examples we have used the function print() to print messages which is built-in python.
		We also used custom created functions to cast our abilities, like ability().
		A function can be created like:
		>>> def aNameOfFunction( ):
		>>> print("Function printing message")
		The function aNameOfFunction() would print "Function printing message" every time it's called.
		Functions can take arguments. The function aNameOfFunction() takes no arguments.
		Code inside a function is indented to indicate it belongs to a certain function. Without indentation code isn't considered part of a function.
		>>> def anotherFunction(message):
		... print(message)
		This time, the functions anotherFunction() expects an argument when called. This means that if we try to execute it like: anotherFunction() we would get an error. We have to pass an argument inside the parentheses of the function like: anotherFunctions("Hello") which would print "Hello".
		We can also give variables as arguments to a function like:
		>>> aVariab=100.0
		>>> anotherFunction(a)
		which will print: 100.0
		That would print "100.0" since it's the value of the variable aVariab we passed as arguement to the function.
			spells once, doesn't mean our ability system is ready.
			We know already that we need to meet certain conditions in order to use a spell. Without building functions we would have to write code with if/else and while loops to check every time we want to cast a spell.
			We have an excellent opportunity to create functions that would do this for us and we simply need to call them and they would execute everything nested in them, that would check the conditions.

		Functions can be created to take multiple arguments and also return data. The following code:
		>>> def customCreatedfunction(number1,number2):
		... return(number1+number2)
		>>> aResult = customCreatedfunction(3,4)
		>>> print(aResult)
		wll print: 7
		This function here expects 2 arguments and returns their sum. Since it returns a value we set the variable aResult to be whatever our function returns and after that we print aResult. Here it should print "7".	

Κώδικας 23: Κώδικας - σχόλια δοκιμασίας 1- επίπεδο 4

```
# List named inventory has been defined
# The variable that holds our experience points is named exp and has been define
d
counter=0

    if i!="healthPot" and i!="manaPot":

    counter+=1
    if counter>100:
        break
```

Κώδικας 24: Ενδεικτική λύση δοκιμασίας 1 - επίπεδο 4

```
counter=0
for i in inventory:
    if i!="healthPot" and i!="manaPot":
        exp+=1
    counter+=1
    if counter>100:
        break
```

Κώδικας 25: Κώδικας - σχόλια δοκιμασίας 2- επίπεδο 4

```
# The name of the dictionary we want to create is: stats
# Prints should have the format: "String: value"
# hpregen and mpregen happen per second therefore we want to print like:
"String:123/sec"
# In the print commands use the %s to refer to the names (keys) since they are S
trings

for name,value in stats.items():
    if name=="Health Regen" or name=="Mana Regen":

    else:
```

Κώδικας 26: Ενδεικτική λύση δοκιμασίας 2 - επίπεδο 4

```
stats = {"Health Points":health,"Mana Points":mana,
"Attack Damage":attackDamage,"Armor":armor,"Health Regen":hpregen,
"Mana Regen":mpregen}
for name,value in stats.items():
    if name=="Health Regen" or name=="Mana Regen":
        print("%s: %d/sec" %(name,value))
    else:
        print("%s: %d" %(name,value))
```

Κώδικας 27: Κώδικας - σχόλια δοκιμασίας 3- επίπεδο 4

```
# Variable mana has our current mana points
# Create a function called: AbilityR() that would take as arguement our mana po
ints and would execute ultimateAbility()
# To execute ultimateAbility() we need to be checking if we have enough mana (7
0 needed) and if abilityCooldown is 0
# Function names are case sensitive as well as arguements and don't forget iden
tation for nested lines!
# Afterwards call the function you created passing as argument our mana points

    if currentMana>=70 and abilityCooldown==0:

    else:
        print("Impossible action")
# Call the function you created below this:
```

Κώδικας 28: Ενδεικτική λύση δοκιμασίας 3 - επίπεδο 4

```
def AbilityR(currentMana):
    if currentMana>=70 and abilityCooldown==0:
        ultimateAbility()
    else:
        print("Impossible action")


# Call the function you created below this:
AbilityR(mana)
```

5.2.5 Περιεχόμενα επιπέδου 5

Πίνακας 7: Εκπαιδευτικό περιεχόμενο επιπέδου 5

	Α/Α	**Επεξήγηση του εκπαιδευτικού περιεχομένου στο παιχνίδι και παραδείγματα**	**Εκφώνηση προβλήματος**
Δοκιμασία	**1**	In Python we have the option of Object Oriented Programming since Python is an Object Oriented Programming language. This is done by creating Classes in Python.Classes may contain variables and functions. After the creation of a class, we can start creating instances of this class which can contain variables and functions. Note here that variables inside a class or an instance are called attributes of that class or instance and functions are called methods. To explain better what an instance and a class is we can think about classes as blueprints or designs of how to create an instance of this class. Instances are the realization of this blueprint or class. To understand this better let's create a class for the Hero: >>> class Hero( ): ... weapon="sword" Now that we have the class for our hero let's create the instance that the player uses:	We have found some potions before and we have stored them in our inventory list as Strings. We have also used them by checking the namings and deciding what kind of potion it is and what will it do. But what if in the future found all sorts of different potions with different uses and attributes

		>>> player=Hero( )
		>>> print(player.weapon)
		This code will print: sword
		Let's rewrite the code of the Hero class to be:
		>>> class Hero():
		... def __init__(self,name,weapon,stats):
		... self.name=name
		... self.weapon=weapon
		... self.stats=stats
		...
		... def action(self):
		... print("My name is %s!" %self.name)
		... print("My weapon is the %s!" %self.weapon)
		...
		... def usePotion(self,potionClass):
		...
		self.stats[potionClass.usage]+=potionClass.amount
		What we have done here is create methods and attributes for our class. The method __init__ is executed automatically every time an instance of this class is initialized and it's called: constructor. This method (like a function would) expects 2 arguments which means the instance has to be given 2 arguments when initialized in this class. The "self" argument refers to the instance that is being constructed and must always be the first argument in every method of the class.
		If we initialize an instance for our player of the class Hero, when the methods run, the "self" parameter will refer to the instance we have created. If we were to create another instance then "self" would be a reference for that instance. In general, we can use self to access variables of the class.
		>>> stats={"Health Points":1000.0,"Mana Points":200.0,"Attack Damage":50.0,"Armor":10}
			they can affect? It would be wiser to create a class for our potions and simply pass the arguments needed like: name, attribute it will affect and amount.

		>>> player=Hero("George","sword",stats)
		>>> player.action( )
		My name is George!
		My weapon is the sword!
		Note that one argument we have given to the instance player is a dictionary that contains our stats.	

	2	Since we talked about classes and objects and our ability to create and use them, another important part of Object Oriented Programming is Inheritance. We can create a class (child class) that would inherit from another (parent class).
		>>> class Animal( ):
		... def __init__(self,type):
		... self.type=type
		...
		... def getType(self):
		... return self.type
		...
		>>> class Pet(Animal):
		... def __init__(self,name,type):
		... super( ).__init__(type)
		... self.name=name
		...
		... def getType(self):
		... print("This is a pet %s named %s %(self.type, self.name))
		...
		>>> a=Animal("wolf")
		>>> b=Pet("Barker","dog")
		>>> print(a.getType( ))
		>>> b.getType( )
		This makes our code reusable and allows us to easily add functionality to a sub-class that has similarities with the parent class but this functionality will not apply to the parent
			Using the same code from previous task for our Hero class, here we want to create a class that would inherit from the Hero class. This class would represent a Helper in our game that would spawn and help us fight for a short time. This HelperHero class will have a set name "helper", same weapons as the Hero Class and it's own dictionary of helperStats. Since, in the code provided we create an instance of HelperHero

		class. The previous page code would have the output: wolf This is a pet dog named Barker Firstly we created a class Animal that would be our parent class and needs only one argument of the type of the animal. It also has one method that would return what type of animal it is. Then we created a class Pet indicating that it inherits from the class Animal with the command: class Pet(Animal) In the constructor for this child class we have used super( ) which allows us to inherit all attributes and methods from the parent class and gives us access to them. With super( ) we could use inside Pet class: print(super( ).getType( )) This would print what type of animal our Pet object/instance is and in our case "dog". But we have one more attribute in our child class which is the pet's name that we have to pass as argument when creating an instance of this child class. Logically, an animal has a type (cat,dog,wolf,lion etc.) but not all animals have names since not all animals are pets which makes sense. We could stop here and afterwards use 2 print commands: print(a.getType( )) and print(b.getType( )) This would result in the use of the parents' class method getType( ) in both cases and only have the output "wolf" and "dog" respectively. By re-defining the method getType( ) inside the child class we have managed to change it's behavior but only for the child class and instead of returning, it automatically prints a message with the name and the type of the pet. This procedure is called overriding and it allows us to change parent methods for our child classes and all subsequent child classes that might be children classes of this child class Pet.	passing only "sword" and helperStats, we need to call the super( ) constructor and provide it with those attributes AND the set name "helper". It also should print "Looking for enemies!" and if they are found nearby it should print "Attacking enemies!" when calling the action( ) method. Finally, HelperHero class won't be able to use potions in order to heal since it has no inventory (or at least we haven't created one for it) and should print "Can't use potions!" if that method is called.

		>>> class Hero(): ... def __init__(self,name,weapon,stats): ... self.name=name ... self.weapon=weapon ... self.stats=stats ... ... def action(self): ... print("My name is %s!" %self.name) ... print("My weapon is the %s!" %self.weapon) ... ... def usePotion(self,potionClass): ... self.stats[potionClass.usage] += potionClass.amount	

	3	We have seen that in Python sometimes execution of code can raise errors. This can happen from coding mistakes made by the programmer or unforeseen events happening during the execution of code that could lead to generation of error.For example we have the code for our Hero Class and we have just found a legendary sword we want to use:
		The following code:
		>>> weaponRarityDamage={"normal":50}
		>>> class Hero:
		... def __init__(self,name,damage):
		... self.name=name
		... self.weaponDamage=damage
		...
		... def action(self):
		... print("My name is %s!" %self.name)
		... print("My weapon does %d damage!" %self.weaponDamage)
		...
		>>>
		player=Hero("George",weaponRarityDamage["legendary"])
			Given the code above, we want to make sure using try-except blocks of code we are prepared to handle the finding of a "legendary" rarity type weapon that does 100 damage running the command: player=Hero("George",weaponsInventory["legendary"])

		>>> player.action()
		will have the result: KeyError: 'legendary'
		This error was raised because we haven't found other legendary weapons and the key:"legendary" doesn't exist in our dictionary:weaponRarityDamage and so we can't have a damage value for that key.
		This is where try-except blocks of code can be useful. Creating a try block of code allows us to "try" executing that block and if it raises error then we execute except to handle that error. For example without defining the variable aVar:
		>>> try:
		... aVar+=1
		... print(aVar)
		... except:
		... aVar=1
		... aVar+=1
		... print(aVar)
		2
		The code in the try block would alone generate an error since aVar isn't defined but using the except block, we are able to handle that error and continue the execution normally.
		>>> weaponRarityDamage={"normal":50}
		>>> class Hero:
		... def __init__(self,name,damage):
		... self.name=name
		... self.weaponRarityDamage=damage
		... print("My name is %s!" %self.name)
		... print("My weapon does %d damage!" %self.weaponRarityDamage)	

	4	Python has several built-in modules. Modules are code that has been created and stored in a file. For example one such module is math and contains already created mathematical functions. We can't access those functions though, first we	Here there is already created and stored a module that will

		need to import the module:
		>>> import math
		Sometime modules have very long names and it is tedious to type it every time we want to access it. So we can import and rename it for our uses to something else:
		>>> import math as m
		Now that we have imported our module we have access to its functions, for example gcd(int1,int2) expects 2 integer arguments and will return their greatest common divisor:
		>>> import math as m
		>>> print(m.gcd(480,10800))
		The code will print: 240
		Modules allow Python to do just about anything you might need with prebuilt functions. There are many built-in modules in Python and we can install others too but we won't be covering that here. You can look up the official documentation of installing Python modules.
		But what if we wrote our own code and then needed some functionality from that code in another project? Do we need to rewrite our code again in our new project? Good news is that we can create and re-use whenever we want our own modules. To do that we simply need to write the code we need, store it and in the future whenever we have need of it, import it to our working project.
		In some tasks before we have created functions for our abilities and to cast spells. These functions could have great reusability in any other game environment that some other Hero needs them for the same purpose. If we store our function as storedFunction:
		>>> def Ability(currentMana,manaNeeded,abilityCooldown):
		... if currentMana>=manaNeeded and abilityCooldown==0:
		... return True
		... else:
			end the game for us. The function we need accepts as argument an Int that represents the worldLevel we are currently at, we don't know what it does or how it does it but we do know that will return True if this is the final level or False if it's not.

		... return False
		Provided we have defined in our new game project the function ultimateAbility(), then we can:
		>>> import storedFunction as s
		>>> currentMana, manaNeeded, abilityCooldown = 200, 100, 0
		>>> if s.Ability(currentMana,manaNeeded,abilityCooldown):
		... ultimateAbility( )
		>>> else:
		... print("Impossible action")
		Then if the conditions we defined in our storedFunction are met, our new hero in our new project will do what we defined his ultimabeAbility() to do. Note here that since in our original function we return either True or False the if statement simply needs to check which one will be returned.	

Κώδικας 29: Κώδικας - σχόλια δοκιμασίας 1- επίπεδο 5

```
# Hero class code is the same as shown above
# We want to create the Potion class
# It will need to be initialized with: name, usedFor and amount
# After that we create an instance for mana Potion and use it on the Hero instance
 player that has been created
# Potion names are: "healthPot", "manaPot"
stats={"Health Points":1000.0,"Mana Points":200.0,"Attack Damage":50.0,"Armor":10}
player=Hero("George","sword",stats)


        self.name=name
        self.usage=usedFor
        self.amount=amount

mpPot=Potion("manaPot","Mana Points",45)
```

Κώδικας 30: Ενδεικτική λύση δοκιμασίας 1 - επίπεδο 5

```
stats={"Health Points":1000.0,"Mana Points":200.0,"Attack Damage":50.0,"Armor":10
}
player=Hero("George","sword",stats)
class Potion:
    def __init__(self,name,usedFor,amount):
        self.name=name
        self.usage=usedFor
        self.amount=amount

mpPot=Potion("manaPot","Mana Points",45)
player.usePotion(mpPot)
```

Κώδικας 31: Κώδικας - σχόλια δοκιμασίας 2 - επίπεδο 5

```
# Hero class code is as shown above
# Methods need overriding
# enemiesNearby becomes True or False by checkforEnemies() if there enemies clo
se to us
enemiesNearby = checkforEnemies()
helperStats={"Health Points":100.0,"Mana Points":20.0,"Attack Damage":10.0,"Arm
or":2}
player=Hero("George","sword",stats)


    def __init__(self,weapon,stats):



        print("Looking for enemies!")
        if enemiesNearby:
              print("Attacking enemies!")


        print("Cannot use potions!")

h=HelperHero("sword",helperStats)
h.usePotion()
h.action()
```

Κώδικας 32: Ενδεικτική λύση δοκιμασίας 2 - επίπεδο 5

```
enemiesNearby = checkforEnemies()
helperStats={"Health Points":100.0,"Mana Points":20.0,"Attack Damage":10.0,"Armo
r":2}
player=Hero("George","sword",stats)

class HelperHero(Hero):
    def __init__(self,weapon,stats):
        super().__init__("helper",weapon,stats)

    def action(self):
        print("Looking for enemies!")
        if enemiesNearby:
              print("Attacking enemies!")

    def usePotion(self):
        print("Cannot use potions!")

h=HelperHero("sword",helperStats)
h.usePotion()
h.action()
```

Κώδικας 33: Κώδικας - σχόλια δοκιμασίας 3 - επίπεδο 5

```
# Use the try except python commands
# The dictionary should be updated to contain a key:value pair of "legendary"=10
0


    player=Hero("George",weaponRarityDamage["legendary"])


    player=Hero("George",weaponRarityDamage["legendary"])
```

Κώδικας 34: Ενδεικτική λύση δοκιμασίας 3 - επίπεδο 5

```
try:
    player=Hero("George",weaponRarityDamage["legendary"])
except:
    weaponRarityDamage["legendary"]=100
    player=Hero("George",weaponRarityDamage["legendary"])
```

Κώδικας 35: : Κώδικας - σχόλια δοκιμασίας 4 - επίπεδο 5

```
# The module stored is named: checkWhetherWeCanEndTheGame
# The function that it contains is named: gameStateCheck() and it expects an ar
gument
# Import the module as end
# Function getCurrentLevel() returns the level of the map we are in as Int


currentLevel=getCurrentLevel()

    endGame()
else:
    print("We are not done yet!")
```

Κώδικας 36: Ενδεικτική λύση δοκιμασίας 4 - επίπεδο 5

```
import checkWhetherWeCanEndTheGame as end
currentLevel=getCurrentLevel()
if end.gameStateCheck(currentLevel):
    endGame()
else:
    print("We are not done yet!")
```

6 Υλοποίηση του παιχνιδιού

6.1 Περιβάλλον υλοποίησης

Το παιχνίδι σοβαρού σκοπού «Journey of the PyVenturer», υλοποιήθηκε στη μηχανή παιχνιδιών Unreal Engine 4, για τους λόγους που αναλύθηκαν στο Κεφάλαιο 3. Οι περισσότερες λειτουργίες του παιχνιδιού, χαρακτήρα, εχθρών, επιπέδων καθώς και γραφικών, ήχου και user interface, αναπτύχθηκαν εντός της μηχανής παιχνιδιών Unreal Engine 4 με χρήση Tools - Editors αλλά και Blueprints που παρέχει.

Τα εργαλεία Editing, επιτρέπουν την ανάπτυξη πολύπλοκων και σύνθετων γραφικά επιπέδων και κόσμων με εύκολο τρόπο. Παρέχονται δυνατότητες επιλογής, παραμετροποίησης και χρήσης ολόκληρων assets ή κομμάτια αυτών. Έτσι, γίνεται ευκολότερη η δημιουργία κόσμων και επιπέδων καθώς και οι λειτουργίες τους. Ακόμα, παρέχονται εργαλεία δημιουργίας και παραμετροποίησης user interface (UI). Επίσης, με τα εργαλεία (Tools) που διαθέτει η μηχανή Unreal Engine 4 μπορεί ο προγραμματιστής να παραμετροποιήσει επιπλέον το υπό ανάπτυξη παιχνίδι σε σχέση με το επίπεδο γραφικών, επίπεδο λεπτομέρειας, κατεύθυνση φωτός και ανέμου καθώς και αντανακλάσεις (Unreal Engine | Features, n.d.).

Με χρήση Blueprints που υπάρχουν για κάθε οντότητα σε ένα πακέτο παιχνιδιού που δημιουργείται, μπορούν να προγραμματιστούν εύκολα όλες οι απαραίτητες λειτουργίες του παιχνιδιού. Το σύστημα των Blueprint δουλεύει με τοποθέτηση και σύνδεση κουτιών (nodes) σε γραφικό επίπεδο (Tools and Editors, n.d.).

Η μηχανή παιχνιδιών Unreal Engine 4 παρέχει και πληθώρα άλλων δυνατοτήτων που δεν παρουσιάζονται όλες σε αυτό το σημείο.

6.2 Γραφικά και ήχοι

Τα γραφικά και η μουσική που χρησιμοποιήθηκαν για την ανάπτυξη του παιχνιδιού διατίθενται δωρεάν είτε από την Epic Games στο Marketplace για πώληση asset για χρήση με την Unreal Engine 4 είτε από δημιουργούς που διανέμουν δωρεάν το υλικό τους στο Marketplace.

Τα γραφικά και η μουσική διατίθενται δωρεάν:

https://www.unrealengine.com/marketplace/

https://www.unrealengine.com/marketplace/en-US/profile/Epic+Games

https://www.unrealengine.com/marketplace/en-US/profile/REXARD

https://www.unrealengine.com/marketplace/en-US/profile/Taylor+Brook+Music

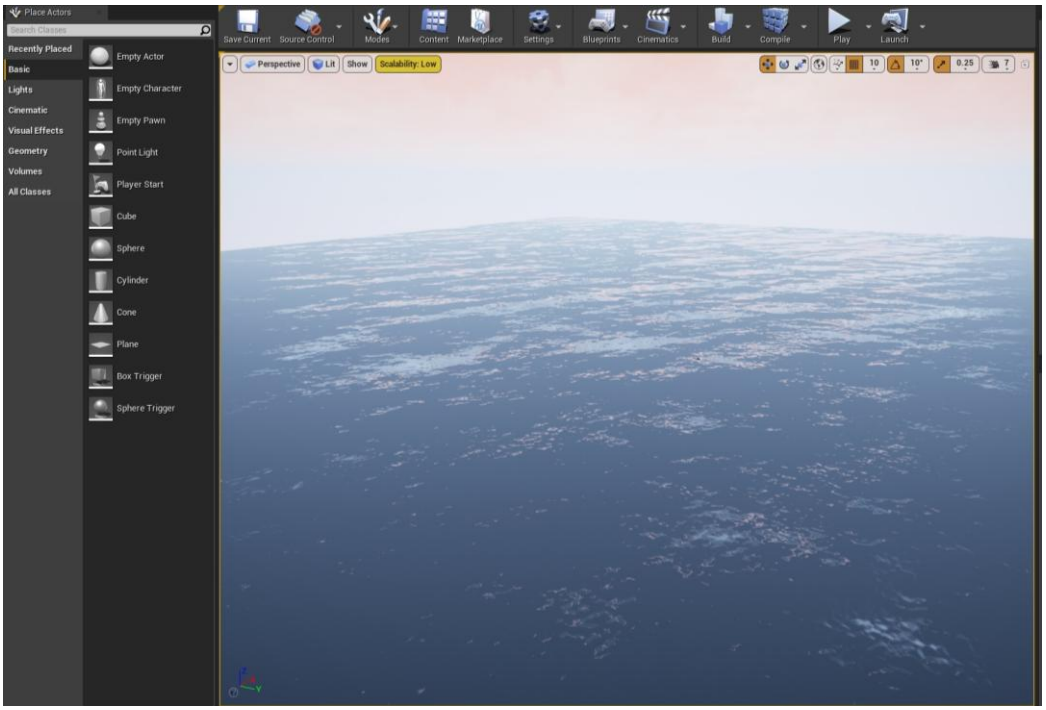
Επίσης, χρησιμοποιήθηκαν ήχοι και ηχητικά εφέ που διανέμονται δωρεάν από:

https://www.zapsplat.com

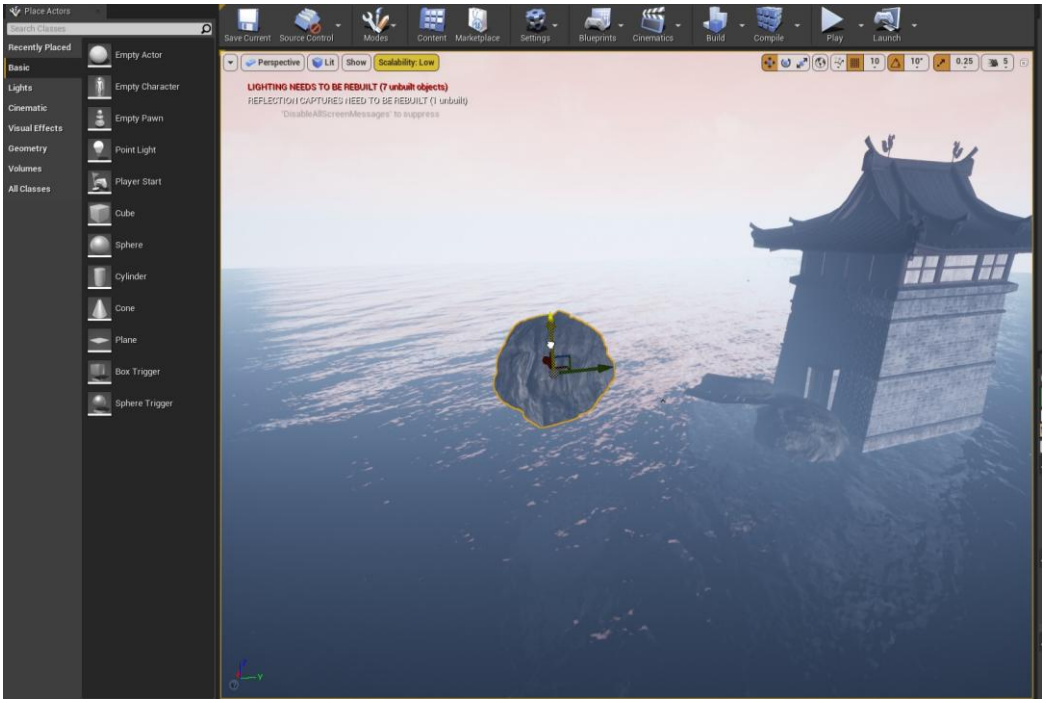
https://sonniss.com/gameaudiogdc

6.3 Δημιουργία επιπέδων

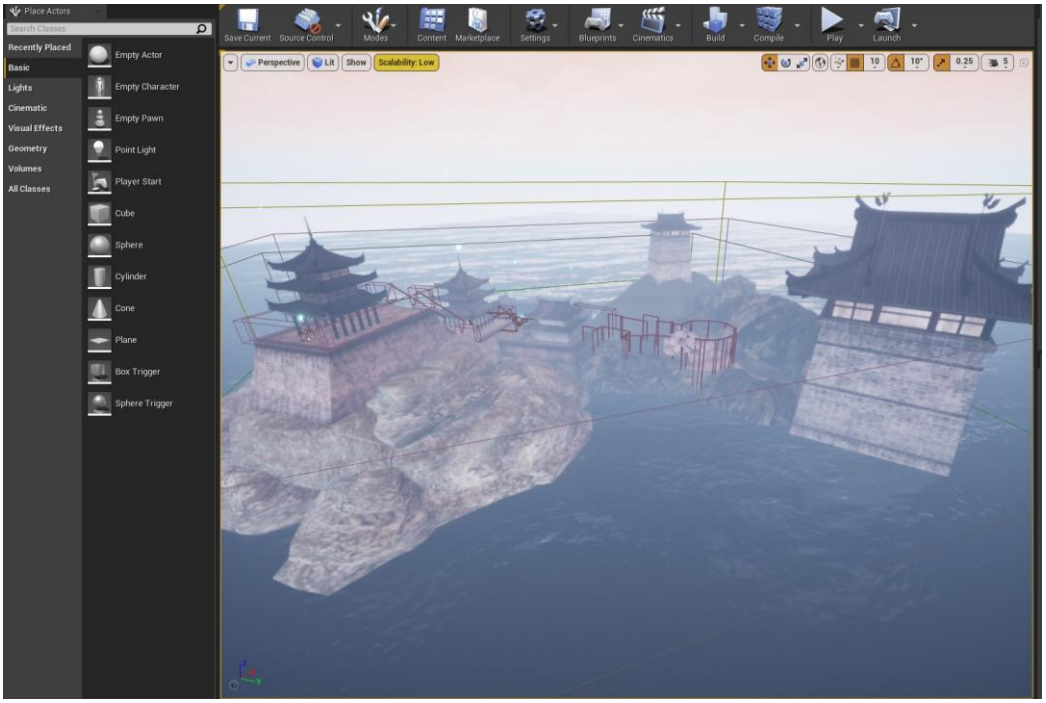
Κατά τη δημιουργία του κάθε επιπέδου, έγινε χρήση assets και blueprints, από τα πακέτα γραφικών που χρησιμοποιήθηκαν. Σε πολλές περιπτώσεις, παρέχονται έτοιμα ολόκληρα Blueprint επιπέδων και δομών με ενσωματωμένα asset, τα οποία χρειάζονται παραμετροποίηση ώστε να εξυπηρετούν τους σκοπούς του παιχνιδιού σοβαρού σκοπού που αναπτύσσεται.



Εικόνα 34: Αρχικό σχέδιο επιπέδου



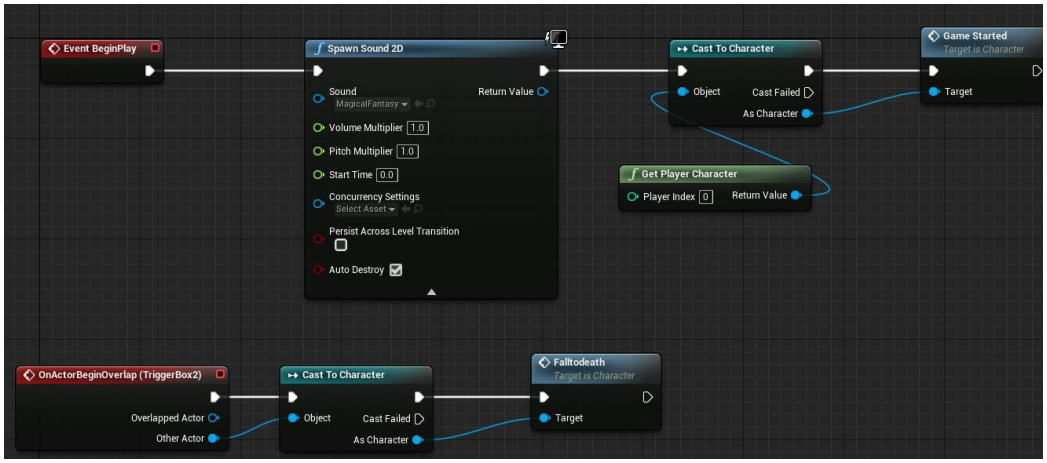
Εικόνα 35: Προσθήκη asset και Blueprint



Εικόνα 36: Τελικό σχέδιο επιπέδου 5

Η διαδικασία που παρουσιάζεται στις Εικόνες 34, 35 και 36 έπρεπε να γίνει για όλα τα επίπεδα του παιχνιδιού.

Σε κάθε επίπεδο επίσης, εκτελούνται με το γεγονός “Event Begin Play” που παρέχει η μηχανή και εκτελείται κατά την εκκίνηση του παιχνιδιού σε ένα επίπεδο, η δημιουργία του μουσικού υπόβαθρου του επιπέδου και η κλήση του event που σχεδιάστηκε στο Blueprint του κεντρικού χαρακτήρα “Game Started” και αναλύεται στη συνέχεια. Ακόμα, σε περίπτωση που ο παίκτης έρθει σε επαφή με ένα “TriggerBox” που βρίσκεται στο επίπεδο της θάλασσας, θεωρείται ότι ο ήρωας έπεσε από τη βασική πλατφόρμα και εμφανίζεται μήνυμα ήττας στην οθόνη του παίκτη.



Blueprint 1: Event Begin Play και Event πτώσης ήρωα

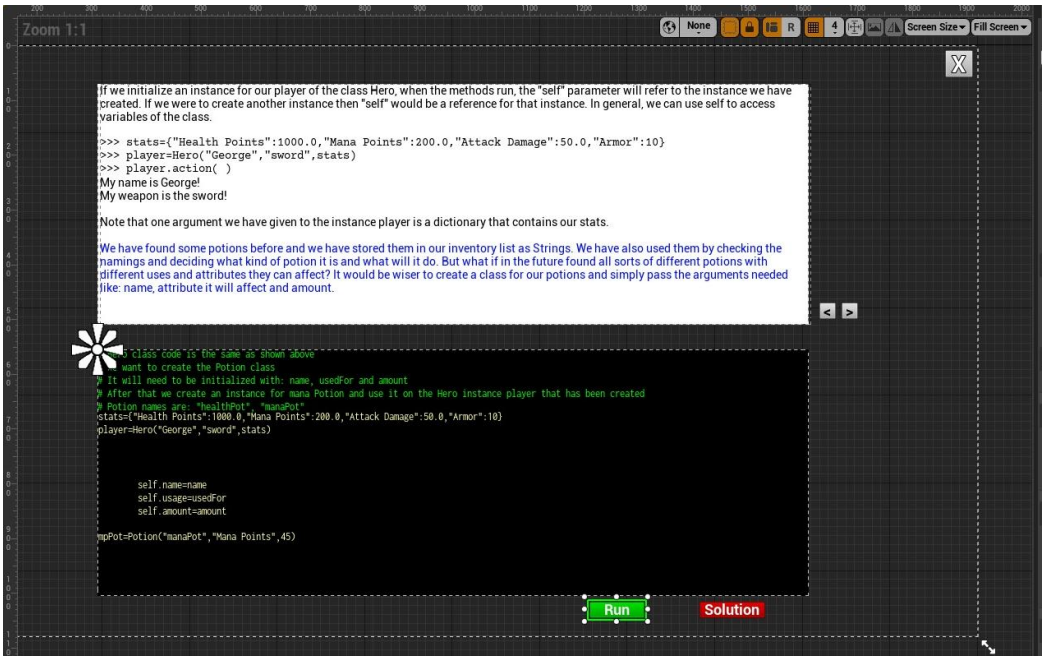
Το βασικό μενού, αποτελεί και αυτό επίπεδο του παιχνιδιού. Η μηχανή, κατά την εκκίνηση, δημιουργεί το επίπεδο 5 του παιχνιδιού. Σε αυτό, έχουν προσαρμοστεί ο χαρακτήρας και κάποια επιπλέον γραφικά στοιχεία που δεν υπάρχουν στο επίπεδο 5 που θα συναντήσει ο παίκτης κατά τη διάρκεια του παιχνιδιού. Επίσης, εμφανίζεται και το UI του κεντρικού μενού όπως παρουσιάζεται στην Εικόνα 5. Τέλος, υπάρχει μια στατική κάμερα και ο παίκτης δεν έχει κάποιον έλεγχο στο παιχνίδι όπως στα άλλα επίπεδα.

6.4 Υλοποίηση User Interface εφαρμογής

Στο παιχνίδι δημιουργήθηκαν και προστέθηκε λειτουργικότητα για διάφορα μενού, όπως φαίνονται στις εικόνες του κεφαλαίου 4 όπου παρουσιάζεται το παιχνίδι. Η δημιουργία τους έγινε με χρήση του Editor που παρέχει η μηχανή. Όλα τα UI της εφαρμογής αποτελούνται από κουμπιά, προσαρμοσμένα στην οθόνη του παίκτη και στο πάτημά τους δημιουργούν κάποιο event. Τα event αυτά, διαχειρίζονται με τρόπο ώστε να δημιουργούν το αποτέλεσμα της επιλογής που αναγράφεται στο κάθε κουμπί. Έτσι, δημιουργούνται τα υπόλοιπα μενού επιλογών “Select Level”, “View Completed Tasks”, “Options” και “Instructions” αλλά και αφαιρούνται και επαναφέρεται το κεντρικό μενού

με την επιλογή “Back” σε κάθε ένα από αυτά. Επίσης, με τις επιλογές “Start new Game”, “Load Game” του κεντρικού μενού και “Open Level” για το μενού “Select Level” γίνεται εκκίνηση του παιχνιδιού στο επίπεδο επιλογής ή φόρτωσης από αποθήκευση. Στην επιλογή “Start new Game”, σε περίπτωση που δεν παίζει ο παίκτης για πρώτη φορά, του εμφανίζεται μήνυμα στην οθόνη αναφέροντας ότι θα χάσει όλες τις αποθηκευμένες καταστάσεις στο παιχνίδι και θα ξεκινήσει από την αρχή αν επιθυμεί να συνεχίσει με νέο παιχνίδι.

User Interface αποτελούν και οι οθόνες των δοκιμασιών που εμφανίζονται στον παίκτη οι οποίες επίσης υλοποιήθηκαν εντός της μηχανής παιχνιδιών Unreal Engine 4.



Εικόνα 37: Δημιουργία οθόνης δοκιμασίας

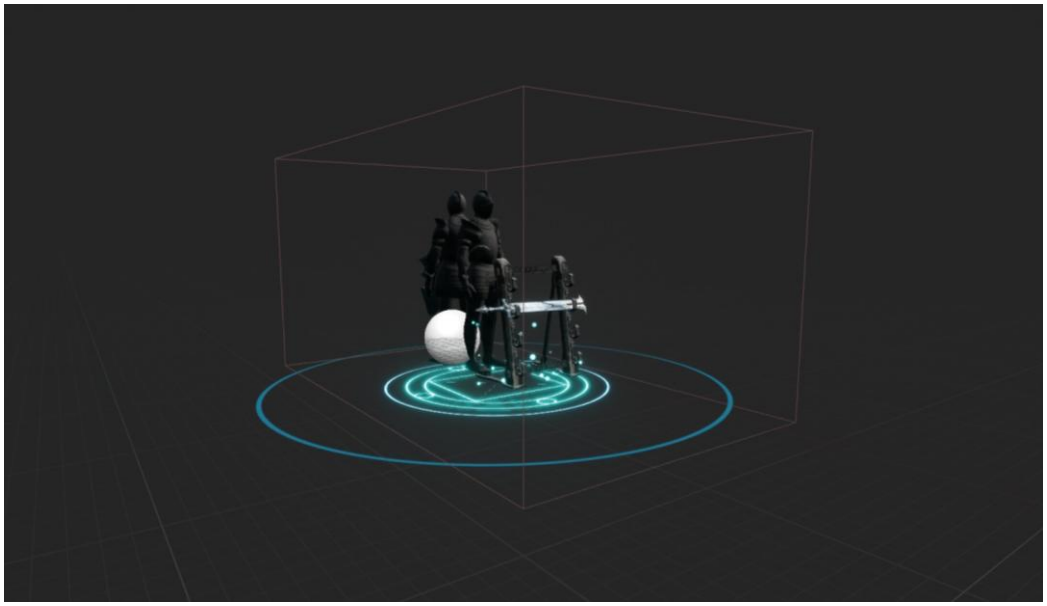
Στην οθόνη της δοκιμασίας ο παίκτης έχει τις επιλογές που αναλύθηκαν στο κεφάλαιο 4 και οι οποίες υλοποιούνται για κάθε δοκιμασία. Η πιο βασική λειτουργία είναι αυτή της εκτέλεσης “Run” της απάντησης του παίκτη και του ελέγχου των αποτελεσμάτων της (Παράρτημα Blueprint 3). Με αυτήν την επιλογή, αρχικά τα περιεχόμενα των απαντήσεων του παίκτη περνάνε στη συνάρτηση “Python Function” που δημιουργήθηκε για το παιχνίδι και θα αναλυθεί στη συνέχεια. Βάσει των αποτελεσμάτων που θα επιστρέψει αυτή η συνάρτηση κρίνεται εάν ο παίκτης έδωσε σωστή απάντηση στη δοκιμασία ή εμφανίζεται μήνυμα λάθους στην οθόνη. Ακόμα και αν ο παίκτης αποφασίσει να χρησιμοποιήσει κάποια από τις βοήθειες που του

παρέχονται, πρέπει να εκτελέσει και πάλι τον κώδικα αλλά αυτή την φορά οι απαντήσεις έχουν αντικατασταθεί με σωστές.

Επίσης, στις δοκιμασίες με εκτενέστερη θεωρία υπάρχουν βελάκια επιλογής σελίδας θεωρίας που με την χρήση ενός μετρητή (counter) είναι υπεύθυνα για την προβολή της επόμενης ή της προηγούμενης σελίδας (Παράρτημα Blueprint 4). Σε αυτές τις περιπτώσεις, ο παίκτης δεν μπορεί να εισάγει κώδικα, καθώς η κονσόλα είναι κρυμμένη, αν δεν δει τουλάχιστον μια φορά όλα τα παράθυρα θεωρίας όπου και εμφανίζουν την κονσόλα.

6.5 Σημεία ενδιαφέροντος

Τα σημεία ενδιαφέροντος στον κόσμο του παιχνιδιού αποτελούν αντικείμενα που φτιάχτηκαν με χρήση Blueprints και κάποιων στατικών πλεγμάτων (static mesh) που δίνονται έτοιμα στα πακέτα γραφικών που χρησιμοποιούνται αλλά και particle textures που δημιουργήθηκαν εντός της μηχανής. Επίσης, προστέθηκε και μια δομή ελέγχου συγκρούσεων (collision box) που δημιουργεί event όταν ο κεντρικός χαρακτήρας έρθει σε επαφή μαζί της. Αυτή η δομή είναι αόρατη και δεν εμποδίζει στην κίνηση τον παίκτη με κάποιο τρόπο.



Εικόνα 38: Σημείο ενδιαφέροντος εξοπλισμού ήρωα

Κατά το event αρχής επικάλυψης του collision box (Event Begin Overlap) γίνεται έλεγχος εάν αυτή η επικάλυψη προέρχεται από το εξάρτημα collision του σώματος του κεντρικού χαρακτήρα και έλεγχος τιμής της Boolean μεταβλητή “Delay” (Παράρτημα

Blueprint 5). Μόνο σε περίπτωση που ο έλεγχος και των δύο παραμέτρων επαληθευθεί, συνεχίζει η ροή εκτέλεσης. Η μεταβλητή “Delay” δημιουργήθηκε διότι με την επιλογή που κλείνει την δοκιμασία από την οθόνη του παίκτη χωρίς αυτός να δώσει λύση, το σημείο ενδιαφέροντος δεν αφαιρείται από τον κόσμο. Όμως, ο χαρακτήρας δεν έχει προλάβει να κινηθεί και να βγει από τον όγκο box collision και έτσι παράγονται εκ νέου event επικάλυψης που ως αποτέλεσμα έχουν την επανεμφάνιση της δοκιμασίας στην οθόνη του παίκτη και επανάληψη της διαδικασίας κάθε φορά που αυτός θα προσπαθεί να την κλείσει, γεγονός που τον οδηγεί σε παγίδευση. Με μια καθυστέρηση ο παίκτης προλαβαίνει να φύγει από τον όγκο που δημιουργεί event χωρίς να παγιδευτεί.

Στην περίπτωση επαλήθευσης των ελέγχων, η δοκιμασία δημιουργείται καλώντας το event “Task” από το Blueprint του κεντρικού χαρακτήρα με Casting στην κλάση του κεντρικού χαρακτήρα και παρέχοντας την αναφορά “self” (Παράρτημα Blueprint 5). Η λειτουργία του event “Task” επεξηγείται σε επόμενη ενότητα. Επίσης υπάρχουν τα event “Waiting” και “Destroy” στο Blueprint του σημείου ενδιαφέροντος. Το πρώτο θέτει και διατηρεί την τιμή της Boolean μεταβλητής “Delay” σε ψευδή για κάποιο χρόνο, για τους λόγο που αναλύθηκε. Το δεύτερο καλείται για να αφαιρέσει το σημείο από τον κόσμο του παιχνιδιού σε περίπτωση που ο παίκτης έδωσε τη σωστή απάντηση. Τα δύο αυτά event θα κληθούν από το αντικείμενο του ήρωα που χειρίζεται ο παίκτης. Το αντικείμενο του ήρωα, γνωρίζει από την αναφορά “self” που του δίνουν τα σημεία ενδιαφέροντος όταν καλούν το event “Task”, για ποιο σημείο ενδιαφέροντος (καθώς υπάρχουν πολλά σε κάθε επίπεδο) θα καλέσει τα event “Waiting” ή “Destroy”.

Η ίδια λειτουργικότητα και λογική υπάρχει σε όλους τους τύπους των σημείων ενδιαφέροντος που ο παίκτης συναντάει στο παιχνίδι.

6.6 Χαρακτήρες και λειτουργικότητα

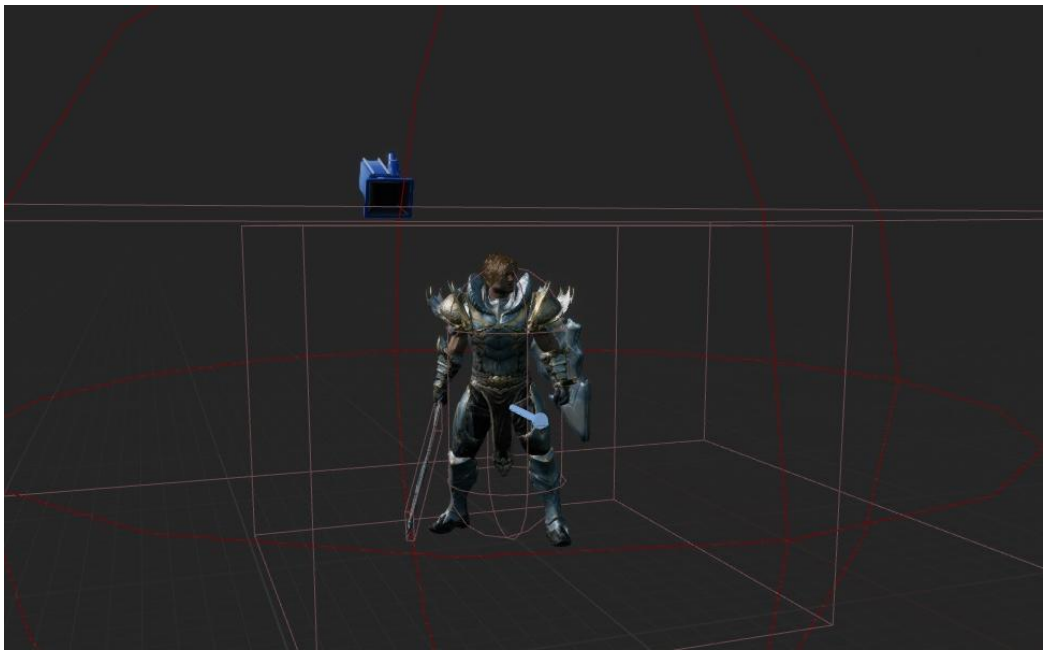
Οι χαρακτήρες που υπάρχουν στο παιχνίδι, κεντρικός χαρακτήρας - ήρωας και εχθροί παρέχονται έτοιμοι με όλες τις κινήσεις (animation) και ηχητικά εφέ. Έπρεπε όμως να προσαρμοστούν στα πλαίσια του παιχνιδιού από πλευράς λειτουργικότητας όσο και από πλευράς κινήσεων.

Όπως αναφέρθηκε, τα Blueprints στην Unreal Engine 4 απαρτίζονται από κουτάκια κόμβων (nodes), τα οποία ενώνονται από τον προγραμματιστή που αναπτύσσει το παιχνίδι για να δημιουργηθεί η ροή κατά την εκτέλεση. Στην συνέχεια

παρουσιάζονται και αναλύονται οι λειτουργίες του κεντρικού χαρακτήρα και των εχθρών.

6.6.1 Κεντρικός χαρακτήρας - ήρωας

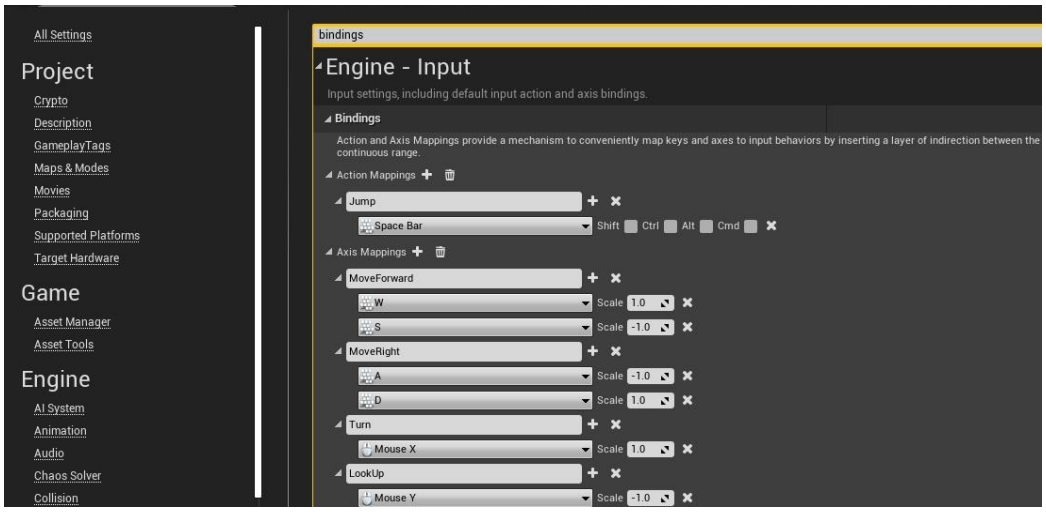
Ο ήρωας που χειρίζεται ο παίκτης απαρτίζεται από το στατικό πλέγμα (static mesh) και τα textures που έρχονται έτοιμα μαζί με αυτόν. Επιπλέον αυτών, προστέθηκαν στον χαρακτήρα διάφορα πλαίσια συγκρούσεων (collision box, collision sphere, capsule collision) που δημιουργούν γεγονότα σύγκρουσης (collision events) και θα βοηθήσουν στον έλεγχο καταστάσεων και στη δημιουργία αποτελεσμάτων. Πίσω και επάνω στον κεντρικό χαρακτήρα υπάρχει τοποθετημένη μια κάμερα που τον ακολουθεί συνεχώς, καθώς είναι προσκολλημένη (attached) μαζί του και αποτελεί την οπτική που θα έχει ο παίκτης καθώς θα παίζει.



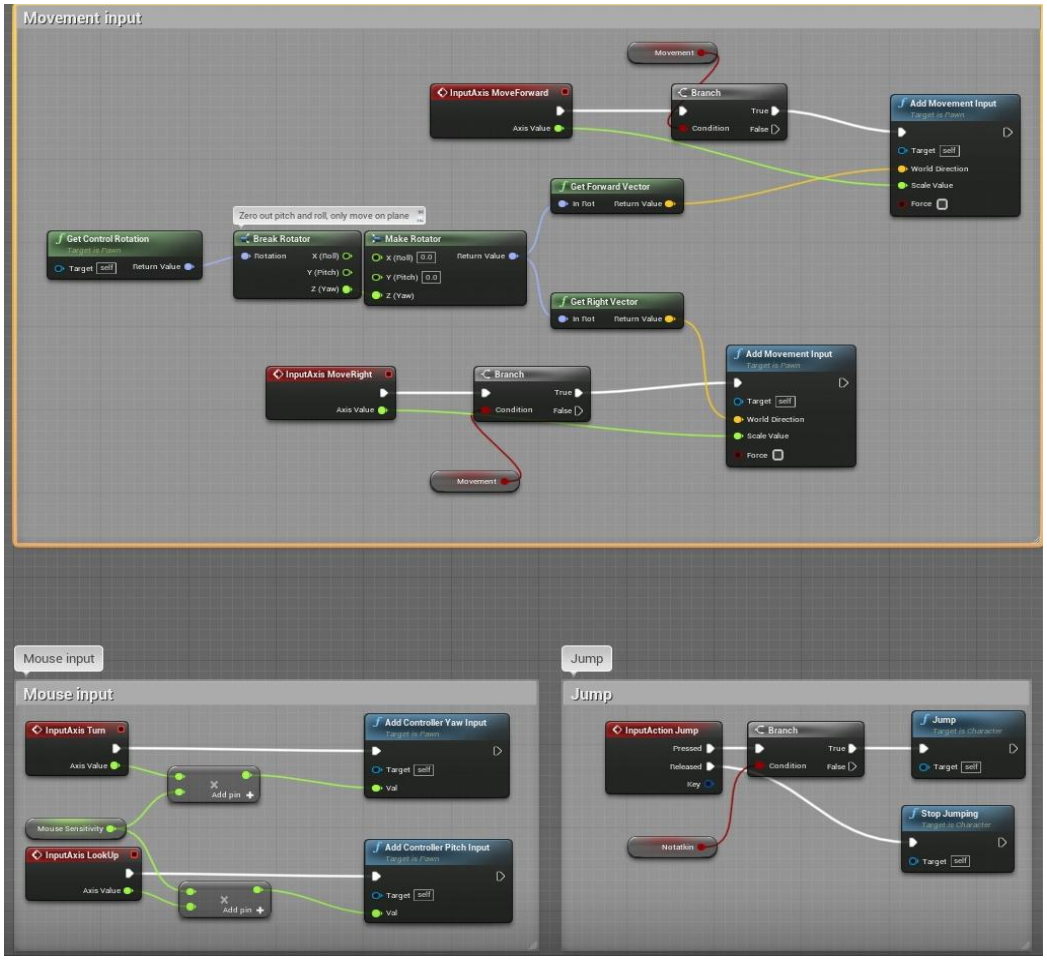
Εικόνα 39: Κεντρικός χαρακτήρας - εξαρτήματα

Στην Unreal Engine 4 υπάρχουν διάφορα γεγονότα (events) που συμβαίνουν κατά την εκτέλεση του παιχνιδιού και ένας δημιουργός μπορεί να τα εκμεταλλευτεί είτε προσθέτοντας στοιχεία που τα δημιουργούν, όπως στον κεντρικό χαρακτήρα με τα στοιχεία συγκρούσεων, είτε ενεργοποιώντας τη λειτουργία που ήδη υπάρχει στην υποκείμενη μηχανή παιχνιδιού.

Αρχικά, η κίνηση του χαρακτήρα υπάρχει ήδη υλοποιημένη από τη μηχανή παιχνιδιών με χρήση event που είναι προγραμματισμένα να παράγουν έργο όταν στη συγκεκριμένη περίπτωση ο παίκτης πατάει κάποιο πλήκτρο ή κάνει κλικ με το ποντίκι ή κινεί το ποντίκι.



Εικόνα 40: Επιλογή πλήκτρων εισόδου και δημιουργία event



Blueprint 2: Χρήση event στην δημιουργία αποτελέσματος

Αυτή η υλοποίηση βασικών κινήσεων παρέχεται έτοιμη από την Unreal Engine 4 μαζί με το Event Graph όπως φαίνεται στο Blueprint 2, αλλά ο προγραμματιστής μπορεί να τα παραμετροποιήσει όπως εξυπηρετεί το έργο του. Στο συγκεκριμένο σημείο έχουν προστεθεί σημεία ελέγχου λογικής if/else (branch εντός Blueprint) όπου οδηγούν στην κίνηση ή όχι βάση μιας λογικής μεταβλητής (Boolean) που χρησιμοποιείται και μεταβάλλεται στη διάρκεια του παιχνιδιού. Επίσης, προστέθηκε πολλαπλασιασμός της εισόδου κίνησης ποντικιού ανάλογα με την επιλογή του παίκτη "Mouse Sensitivity" που υπάρχει στο μενού "Options".

Έχει δημιουργηθεί σειρά λειτουργιών με event το οποίο χρησιμοποιεί το έτοιμο event node που παρέχεται και σε περίπτωση εισόδου αριστερού κλικ με το ποντίκι καλείται (Παράρτημα Blueprint 6). Εδώ, βρίσκεται η λειτουργία βασικής επίθεσης του χαρακτήρα. Αρχικά εξετάζονται η μεταβλητή Boolean "HasSword" που ορίζεται True όταν ο παίκτης λύσει τη δοκιμασία εξοπλισμού στο δεύτερο επίπεδο και η Boolean επιστροφή της συνάρτησης "is falling" της μηχανής ώστε να μην μπορεί ο ήρωας να επιτεθεί αν δεν έχει όπλα ή αν βρίσκεται σε ελεύθερη πτώση. Εάν οι συνθήκες επαληθεύονται τότε ξεκινάει η λειτουργία animation της επίθεσης, μεταβολές σε μεταβλητές που ελέγχουν σε ποιο σημείο της κίνησης μπορεί να παράγει ο παίκτης ζημία (damage) και δημιουργούνται τα κατάλληλα ηχητικά εφέ.

Επίσης, με χρήση ενός από τα εξαρτήματα collision που ακολουθεί το σπαθί, εξετάζεται εάν αυτό έρχεται σε σύγκρουση με κάποιον εχθρό ώστε να παράγει damage.

Με παρόμοιο τρόπο εξετάζεται και η άμυνα του ήρωα με χρήση ασπίδας στη μάχη με τους εχθρούς. Η μοναδική διαφορά αποτελεί η κλήση του event "Deflecting" που έχει δημιουργηθεί ώστε ο ήρωας να μην δεχτεί damage, για μικρό χρονικό διάστημα που διαρκεί το block animation, σε περίπτωση που ο παίκτης κάνει δεξί κλικ με το ποντίκι (Παράρτημα Blueprint 7).

Με όμοιο αλλά ποιο σύνθετο τρόπο, ελέγχεται και χρησιμοποιείται και η πρώτη ικανότητα του χαρακτήρα. Γίνεται έλεγχος των βασικών Boolean μεταβλητών που έχουν προστεθεί στο παιχνίδι αλλά καλείται επιπλέον και η συνάρτηση Spell Check που δημιουργήθηκε ώστε να ελέγχει εάν ο παίκτης έχει αρκετούς πόντους mana και αν η ικανότητα έχει χρόνο επαναφόρτισης (cooldown). Αυτή η συνάρτηση δέχεται ως είσοδο το κόστος mana της ικανότητας και την μεταβλητή χρόνου cooldown της ικανότητας και θέτει την Boolean μεταβλητή "do/don't ability" σε True ή False εάν μπορεί να

χρησιμοποιηθεί η ικανότητα ή όχι. Τα επόμενα βήματα είναι ίδια με τις βασικές επιθέσεις αλλά, όπως εξηγείται και στον παίκτη εντός παιχνιδιού, αυτή η ικανότητα μένει ενεργοποιημένη όσο ο ήρωας έχει αρκετό mana ή δεν πατήθηκε ξανά το πλήκτρο ώστε να απενεργοποιηθεί.

Έτσι, όσο ο παίκτης δεν πατάει το πλήκτρο απενεργοποίησης, η ικανότητα καλεί συνεχώς την ενεργοποίησή της και ελέγχει εάν έχει αρκετό mana ή αν ο παίκτης πάτησε το πλήκτρο προς απενεργοποίηση όπου και τελικά σταματάει η εκτέλεση της ικανότητας αυτόματα.

Η δεύτερη ικανότητα έχει ίδια λειτουργία με τις βασικές επιθέσεις αλλά διαφορετικά animation και καλεί όπως η πρώτη ικανότητα τη συνάρτηση “Spell Check” και θέτει τις μεταβλητές που είναι απαραίτητες στις σωστές τιμές. Έπειτα από αυτό τερματίζει καθώς είναι ικανότητα που όταν χρησιμοποιείται παράγει αποτέλεσμα μια φορά έως ότου να μπορεί να επαναχρησιμοποιηθεί.

Ο παίκτης μπορεί οποιαδήποτε στιγμή κατά την διάρκεια του παιχνιδιού, να “παγώσει” (pause) τη λειτουργία του, εμφανίζοντας το μενού “Pause” (Παράρτημα Blueprint 8). Καθώς αυτή η λειτουργία γίνεται με βάση την απόφαση του παίκτη, είναι λογικό να τη διαχειρίζεται η κλάση του κεντρικού χαρακτήρα που ελέγχει ο παίκτης. Έτσι, με πάτημα του πλήκτρου Escape από τον παίκτη καλείται η “Toggle Pause” που δημιουργήθηκε για να οδηγεί σε ροή “Flip Flop”. Εκεί αυτόματα, από την μηχανή, εκτελείται η ροή A ή ροή B βάσει ποιας εκτελέστηκε τελευταία, ξεκινώντας από την A. Στην ροή A, το παιχνίδι γίνεται paused, δημιουργείται το κατάλληλο widget και προστίθεται στην οθόνη. Ενεργοποιείται η εμφάνιση του κέρσορα για το ποντίκι και εκτελείται το animation “παγώματος” του παιχνιδιού καθώς και η αποδοχή εισόδων στο παιχνίδι μόνο με βάση κάποιο UI. Στην ροή B, συμβαίνει το αντίθετο. Αφαιρείται το widget από την οθόνη, αποκρύπτεται ο κέρσορας και συνεχίζει η ροή παιχνιδιού δεχόμενη εισόδους μόνο λειτουργίας και χειρισμού παιχνιδιού (κίνηση, επίθεση, άμυνα, ικανότητες).

Με όμοιο τρόπο δημιουργείται και η εμφάνιση της σελίδας στατιστικών και inventory του χαρακτήρα αλλά με τη διαφορά ότι το παιχνίδι δεν γίνεται Paused και υπάρχει δυνατότητα χρήσης εισόδων από το UI με επιλογή αντικειμένου με το ποντίκι από το inventory και απόκρυψης της σελίδας με χρήση πλήκτρου που έχει λειτουργία στη ροή παιχνιδιού. Με την επιλογή και χρήση κάποιου αντικειμένου τύπου φίλτρου, στον κεντρικό χαρακτήρα, καλείται το event που έχει δημιουργηθεί και υλοποιεί αυτή τη

λειτουργία (Παράρτημα Blueprint 9). Αρχικά, ανάλογα με τον τύπο αντικειμένου φίλτρου που επιλέχθηκε φίλτρο ζωής ή φίλτρο mana (0 ή 1 αντίστοιχα), ακολουθείται διαφορετική ροή. Σε κάθε ροή όμως, ανάλογα με τη χρήση που έχει οριστεί, αυξάνονται οι πόντοι ζωής ή mana με καθυστέρηση ενός δευτερολέπτου για 4 και 2 δευτερόλεπτα αντίστοιχα. Για την αύξηση των πόντων καλούνται τα event που έχουν δημιουργηθεί "Heal" και "MPregen" όπου αυξάνουν την τιμή health ή mana αντίστοιχα όταν κληθούν κατά συγκεκριμένο επίπεδο και φροντίζουν να διατηρούνται οι τιμές των μεταβλητών health και mana μέσα στα όρια μέγιστου εύρους τους που έχουν οριστεί.

Ανάλογα, με το σημείο ενδιαφέροντος με το οποίο ήρθε σε επαφή ο ήρωας, τον αριθμό των δοκιμασιών που έχει λύσει και το επίπεδο στο οποίο βρίσκεται, όταν κληθεί το event "Task" που έχει δημιουργηθεί, εμφανίζεται στην οθόνη και το αντίστοιχο widget δοκιμασίας (Παράρτημα Blueprint 10). Έπειτα καλείται το event "Toggle" που δημιουργήθηκε για να εμφανίζει ή εξαφανίζει από την οθόνη του παίκτη τη δοκιμασία (Παράρτημα Blueprint 11). Στο event "Toggle" ακολουθείται ίδια διαδικασία με τη δημιουργία του μενού "Paused" καθώς δεν θα ήταν λογικό κατά τη διάρκεια που ο παίκτης διαβάζει την θεωρία ή σκέφτεται τη σωστή απάντηση ή εισάγει κώδικα, να δέχεται ζημιά από εχθρούς με αποτέλεσμα να οδηγηθεί σε ήττα "Game Over". Στην δεύτερη ροή του "Flip Flop" στο event "Toggle" ελέγχεται αρχικά αν ο παίκτης απάντησε ή απλώς έκλεισε τη δοκιμασία. Εάν απλά έκλεισε το UI της δοκιμασίας τότε αυτή αφαιρείται από την οθόνη του. Σε περίπτωση που δόθηκε σωστή απάντηση, τότε καλούνται τα event "Destroying" και "Leveling" ώστε να αφαιρεθεί το σημείο ενδιαφέροντος από τον κόσμο του παιχνιδιού και να αυξηθεί το επίπεδο ικανοτήτων του ήρωα και έπειτα αφαιρείται από την οθόνη του παίκτη το UI της δοκιμασίας. Στη συνέχεια, καλείται το event "Delaying" και μετά από 1 δευτερόλεπτο το event "Checker".

Στο event "Checker", ελέγχεται ανάλογα με το επίπεδο αν ο παίκτης απάντησε και την τελευταία δοκιμασία του επιπέδου σωστά (Παράρτημα Blueprint 11). Σε αυτή την περίπτωση, η κατάσταση του παιχνιδιού αποθηκεύεται ("Save Game") όπως και το επίπεδο που βρίσκεται ο παίκτης ("Save Level") και αν δεν είναι το τελευταίο επίπεδο του παιχνιδιού, μεταβαίνει στο επόμενο.

Στα event “Destroying” και “Delaying” καλούνται τα αντίστοιχα event των σημείων ενδιαφέροντος, ανάλογα με το σημείο στο οποίο βρίσκεται ο χαρακτήρας.

Κάθε επίπεδο του παιχνιδιού, μπορεί να καλέσει με casting τα event του χαρακτήρα “Game Started” και “FalltoDeaht”. Η πρώτη, ενημερώνει το αντικείμενο του ήρωα που δημιουργήθηκε λόγο αλλαγής επιπέδου, ώστε να επαναφέρει στον χαρακτήρα τα αποθηκευμένα δεδομένα πριν αλλάξει το επίπεδο (Παράρτημα Blueprint 12).

Οτιδήποτε σχεδιάζεται με Blueprint αποτελεί μια κλάση με δικές της ιδιότητες, μεθόδους και λειτουργία. Όταν τοποθετείται στον κόσμο των επιπέδων ένα αντικείμενο με βάση κάποιο Blueprint, αποτελεί ουσιαστικά ότι και ένα αντικείμενο σε ένα πρόγραμμα μιας αντικειμενοστρεφούς γλώσσας προγραμματισμού και με το κλείσιμο του επιπέδου, ακόμα και για άνοιγμα νέου επιπέδου, χάνεται. Έτσι είναι απαραίτητο να επαναφέρουμε στον χαρακτήρα του ήρωα τα στατιστικά που είχε πριν αλλάξει επίπεδο, διαφορετικά θα ξεκινήσει στο επόμενο επίπεδο με τα χαρακτηριστικά που ορίζονται στο Blueprint του χαρακτήρα (κλάση του ήρωα) και είναι τα βασικά που δίνονται κατά την πρώτη δημιουργία παιχνιδιού “Start new Game” από το κεντρικό μενού.

Επίσης, δημιουργούνται τα κατάλληλα Widget που είναι απαραίτητα και εισάγονται στην οθόνη του παίκτη. Το Widget που έχει τη σελίδα στατιστικών και το inventory του παίκτη, είναι από την αρχή του παιχνιδιού στην οθόνη αλλά ενεργοποιούνται οι επιμέρους λειτουργίες του, μόνο όταν ο παίκτης απαντήσει σωστά στις σχετικές δοκιμασίες. Τέλος, ξεκινάει να λειτουργεί και να καλείται κάθε ένα δευτερόλεπτο, η συνάρτηση “Passive Regen” μέσω του event “Regen” και στην οποία αυξάνονται τα επίπεδα ζωής και mana του ήρωα αλλά και μειώνονται οι χρόνοι cooldown των ικανοτήτων του.

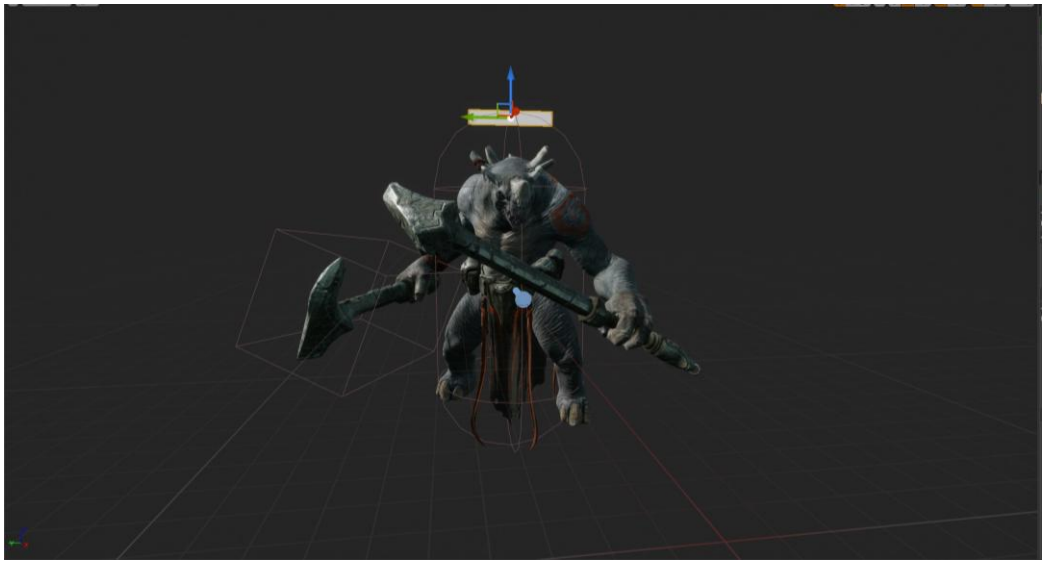
Στο event “FalltoDeath”, όταν κληθεί από κάποιο επίπεδο καθώς ο παίκτης ήρθε σε επαφή με τα αντικείμενα “Trigger” που ορίζουν ότι έπεσε στο κενό, τότε δημιουργείται και προβάλλεται με το κατάλληλο ηχητικό εφέ η οθόνη ήττας “Game Over” (Παράρτημα Blueprint 12).

Στα event “SaveLevel” και “LoadGame” που χρησιμοποιούνται από άλλα events και συναρτήσεις, ελέγχεται αρχικά αν υπάρχει ήδη αποθηκευμένο παιχνίδι (Παράρτημα Blueprint 13). Στο πρώτο, σε περίπτωση που βρεθεί αποθήκευση, ελέγχεται έπειτα αν το επίπεδο (level) που βρίσκεται ο παίκτης είναι μεγαλύτερο από το μέγιστο που υπάρχει αποθηκευμένο και τελικά αποθηκεύεται το μεγαλύτερο. Το “LoadGame” ελέγχει την

ύπαρξη αποθήκευσης καθώς θα δημιουργήσει σφάλμα εάν προσπαθήσει να επαναφέρει το παιχνίδι από αποθηκευμένη κατάσταση ενώ δεν υπάρχει τέτοια. Το event “saveGame” δημιουργεί κάθε φορά νέο αντικείμενο τύπου “Save Game” διαγράφοντας το προηγούμενο εάν υπάρχει καθώς αποτελεί την τελευταία κατάσταση που βρισκόταν ο παίκτης. Εφόσον, το “saveLevel” διατηρεί το μέγιστο επίπεδο που έχει φτάσει ο παίκτης και τα χαρακτηριστικά (health, mana, exp, level, damage, armor) πρέπει να μεταφέρονται σε κάθε άνοιγμα νέου επιπέδου, όπως αναλύθηκε, δεν είναι απαραίτητη η διατήρηση της προηγούμενης αποθήκευσής τους. Τέλος, καθώς το event “LoadLevel” καλείται μόνο από τις επιλογές των μενού, είναι διασφαλισμένο ότι θα υπάρχει αποθηκευμένη κατάσταση από το γεγονός ενεργοποίησης των επιλογών “Load Game” και “Select Level” στο κεντρικό μενού ή όχι.

6.6.2 Εχθροί

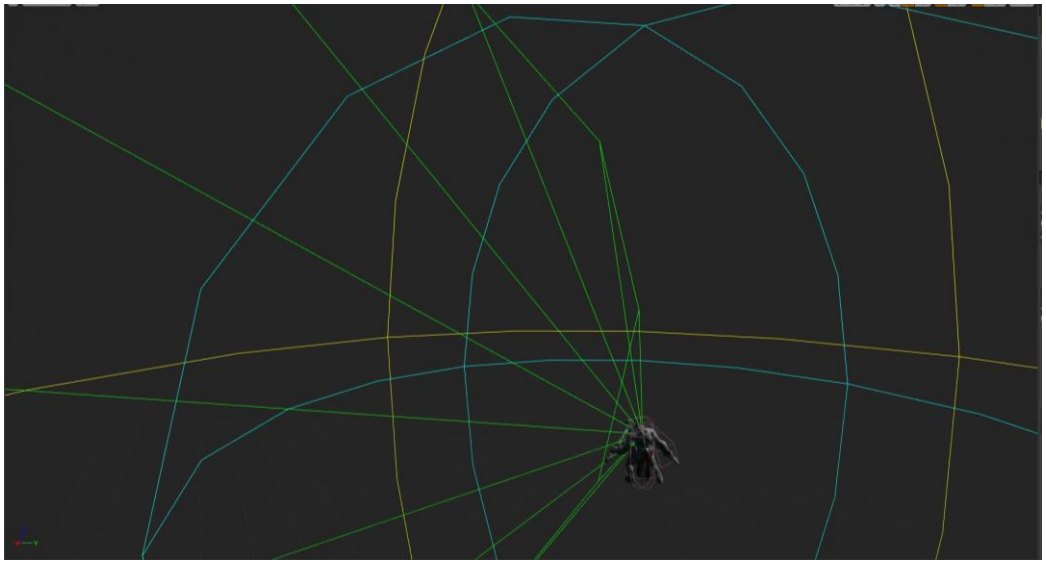
Οι εχθροί όπως και ο κεντρικός χαρακτήρας απαρτίζονται από στατικά πλέγματα τα γραφικά των οποίων προέρχονται από τα πακέτα γραφικών που χρησιμοποιήθηκαν. Επάνω τους έχουν προστεθεί capsule collision και box collision εξαρτήματα που δημιουργούν event σύγκρουσης. Το πρώτο χρησιμοποιείται για να βρεθεί αν το collision box του σπαθιού του παίκτη δημιούργησε event σύγκρουσης με το capsule collision που περιβάλλει το σώμα του εχθρού ώστε να του προκληθεί ζημιά (damage). Το δεύτερο, περιβάλει και ακολουθεί τις κινήσεις των όπλων που δημιουργούνται από τα animation επίθεσης των εχθρών και εξετάζει αν ήρθε σε επαφή με το capsule collision του σώματος του ήρωα ώστε να προκληθεί ζημιά σε αυτόν. Ακόμα, προστέθηκε ένα απλό widget μπάρας πάνω από το κεφάλι του εχθρού όπου το επίπεδο της μπάρας συνδέεται με το επίπεδο ζωής του και γίνεται render σε οπτική κόσμου τριών διαστάσεων σε σχέση με τη θέση της κάμερας του παίκτη. Τέλος, προστέθηκε ένα εξάρτημα τύπου “PawnSensing” που παρέχεται έτοιμο λειτουργικά από τη μηχανή.



Εικόνα 41: Εχθρός τύπου 1

Αρχικά, κατά την εκκίνηση κάθε επιπέδου αυξάνεται η δυσκολία του εχθρού πολλαπλασιαστικά βάσει του επιπέδου που βρίσκεται με αύξηση των πόντων ζωής του καθώς και του επιπέδου ζημιάς που μπορεί να προκαλέσει. Στο event “KnockBack”, το οποίο καλείται από το αντικείμενο του ήρωα, για κάθε αντικείμενο τύπου εχθρού που βρίσκεται εντός του collision εξαρτήματος της δεύτερης ικανότητας του ήρωα, δημιουργείται δύναμη εκτόξευσης του εχθρού σε αντίθετη κατεύθυνση από τον ήρωα. Το event σύγκρουσης του collision box που ακολουθεί τα όπλα του εχθρού, λειτουργεί με τον ίδιο τρόπο με αυτό του ήρωα (Παράρτημα Blueprint 14).

Το εξάρτημα τύπου “PawnSensing” που προστέθηκε, δημιουργεί για τον εχθρό event σε περίπτωση που βρεθεί στο οπτικό του πεδίο κάποιο συγκεκριμένο αντικείμενο κλάσης που ορίζεται (Παράρτημα Blueprint 15). Για τις κλάσεις των εχθρών αυτό μπορεί να είναι μόνο ο ήρωας και με τη δημιουργία τέτοιου event, η μεταβλητή της περιφερειακής όρασης λαμβάνει τη μέγιστη τιμή μοιρών (180^{o}) ώστε από τη στιγμή που ένας εχθρός έχει δει τον ήρωα, να είναι δύσκολο να σταματήσει να τον κυνηγάει. Στη συνέχεια ο εχθρός κινείτα προς την τοποθεσία του ήρωα που είδε και μόλις φτάσει σε αυτόν, ξεκινάει την επίθεση με την ίδια λογική και ελέγχους που κάνει η κλάση του ήρωα όταν επιτίθεται.



Εικόνα 42: Αρχικό οπτικό πεδίο εχθρού

Οι εχθροί δεύτερου τύπου υλοποιούνται και λειτουργούν με τον ίδιο τρόπο που αναλύθηκε για τους εχθρούς τύπου 1.

6.7 Ενσωμάτωση Python

Όπως αναφέρθηκε, η μηχανή Unreal Engine 4 παρέχει στον προγραμματιστή τη δυνατότητα να δημιουργήσει κώδικα σε C++ που θα εξυπηρετεί τους σκοπούς του παιχνιδιού και να επεκτείνει την λειτουργικότητά της. Επίσης, η Python παρέχει βιβλιοθήκες επέκτασης και ενσωμάτωσής σε εφαρμογές C και C++.

Συνδυάζοντας τα παραπάνω, υλοποιήθηκε μια συνάρτηση, η οποία μπορεί να κληθεί μέσα από το περιβάλλον Blueprint της μηχανής και θα ελέγχει τον κώδικα που εισάγει ο παίκτης. Η υλοποίηση βασίστηκε στο "plug-in" που έχει αναπτυχθεί και επεκτείνει την λειτουργικότητα της μηχανής με χρήση Python, τόσο σε επίπεδο δημιουργίας και σχεδίασης όσο και κατά την λειτουργία του παιχνιδιού (https://github.com/20tab/UnrealEnginePython).

Αρχικά, στη συνάρτηση, μεταβιβάζεται ο κώδικας που αναπτύσσει ο παίκτης σε μια δοκιμασία και έπειτα καλώντας λειτουργίες σε Python που υλοποιήθηκαν για τον έλεγχο του κώδικα κάθε δοκιμασίας, επιστρέφει το σφάλμα (error) που προκύπτει ή εάν δόθηκε σωστή απάντηση.

Με αυτόν τον τρόπο, υπάρχει δυνατότητα ελέγχου του κώδικα του παίκτη για βασικά συντακτικά λάθη από τον interpreter της γλώσσας Python και επιστροφή του σφάλματος που προέκυψε. Αυτό, εκτός από το γεγονός διευκόλυνσης των ελέγχων, παρέχει στον παίκτη την αίσθηση δημιουργίας κώδικα σε πραγματικό περιβάλλον ανάπτυξης. Επίσης, με χρήση διάφορων module που παρέχει η Python μπορούν να ελεγχθούν και άλλες παράμετροι του κώδικα του παίκτη που δεν επιστρέφουν συντακτικό σφάλμα αλλά είναι λάθη που σχετίζονται με τα ζητούμενα της εκάστοτε δοκιμασίας.

Κώδικας 37: Header της συνάρτησης (Blueprint Function Libraries, n.d.)

```
UCLASS()
class JOURNEYOFPYVENTURER_API UMyPythonFunctions : public UBlueprintFunctionLibrary
{
    GENERATED_BODY()
        UFUNCTION(BlueprintCallable, Category = "PythonFunctions")
        static FString pythonFunction(FString String1, FString String2,
FString String3, FString task);
};
```

Κώδικας 38: C++ που καλεί Python

```
Py_Initialize();
PyObject* test_Task = PyImport_ImportModule(TCHAR_TO_UTF8(*task));
PyObject* main_func = PyObject_GetAttrString(test_Task, (char*)"main");
PyObject* test1 = PyBytes_FromString(TCHAR_TO_UTF8(*String1));
PyObject* test2 = PyBytes_FromString(TCHAR_TO_UTF8(*String2));
PyObject* test3 = PyBytes_FromString(TCHAR_TO_UTF8(*String3));
PyObject_CallFunctionObjArgs(main_func, test1, test2, test3, NULL);
Py_Finalize();

Py_Initialize();
PyObject* test_Task2 = PyImport_ImportModule(TCHAR_TO_UTF8(*task));
PyObject* main_func2 = PyObject_GetAttrString(test_Task2, (char*)"main2");
PyObject* ret = PyObject_CallFunction(main_func2, NULL);
return (const char*)PyUnicode_AsUTF8(ret);
Py_Finalize();
```

Κρίθηκε σημαντικό, να μην είναι απαραίτητο για τον παίκτη να έχει ή να χρειάζεται να εγκαταστήσει κάποια έκδοση Python. Στην περίπτωση που ο παίκτης έχει στο σύστημά του κάποια τοπική εγκατάσταση Python, ιδιαίτερα αν είναι πιο νέας έκδοσης από αυτή που καλεί η συνάρτηση στο παιχνίδι, έχει προβλήματα εκτέλεσης και λειτουργίας του παιχνιδιού. Για τον λόγο αυτό συμπεριλαμβάνεται στα αρχεία του παιχνιδιού το πακέτο Python embed που αποτελεί μια βασική και φορητή έκδοση της Python και μπορεί να λειτουργήσει σε οποιοδήποτε σύστημα υποστηρίζεται από την έκδοσή της.

Καθώς παρατηρήθηκαν προβλήματα με διάφορα συστήματα και εκδόσεις Python, επιλέχθηκε τελικά η έκδοση Python embed 3.7.9. Επίσης, στη βιβλιοθήκη προστέθηκε και ο κώδικας σε Python που δημιουργήθηκε για κάθε δοκιμασία και τον έλεγχό της.

Αρχικά καλείται, ανάλογα με τη δοκιμασία, η πρώτη συνάρτηση στο αντίστοιχο module Python που έχει δημιουργηθεί για τη συγκεκριμένη δοκιμασία.

Κώδικας 39: Λειτουργία πρώτης συνάρτησης Python

```
import os

def main(answer1,answer2,answer3):
    path = os.path.dirname(__file__)

    fname = "playerCode.py"
    answer1 = answer1.decode("utf-8")
    answer2 = answer2.decode("utf-8")
    answer3 = answer3.decode("utf-8")
    answer1 = answer1+"\n"+answer2
    with open(path+'/'+fname,'w+') as f:
        f.write("class Hero():\n")
        f.write("    def __init__(self,name,weapon,stats):\n")
        f.write("        self.name=name\n")
        f.write("        self.weapon=weapon\n")
        f.write("        self.stats=stats\n")
        f.write("    def action(self):\n")
        f.write("        print('My name is %s!' %self.name)\n")
        f.write("        print('My weapon is the %s!' %self.weapon)\n")
        f.write("    def usePotion(self,potionClass):\n")
        f.write("        self.stats[potionClass.usage]+=potionClass.amount\n")
        f.write(answer1)
        f.write("\n        self.name=name\n        self.usage=usedFor\n
 self.amount=amount\n")
        f.write("def theAnswer():\n")
        f.write("    stats={'Health Points':1000.0,'Mana Points':200.0,
'Attack Damage':50.0,'Armor':10}\n")
        f.write("    player=Hero('George','sword',stats)\n")
        f.write("    mpPot=Potion('manaPot','Mana Points',45)\n")
        f.write("    " + answer3)
```

Ο κώδικας που εισάγει ο παίκτης αποθηκεύεται σε ένα αρχείο τοπικά. Μαζί με τον κώδικα του παίκτη, αποθηκεύονται και όλα όσα θεωρεί η εκφώνηση και η δοκιμασία δεδομένα για το πρόβλημα. Τέλος, τα σημαντικότερα σημεία που πρέπει να εξετασθούν για την δοκιμασία, έχουν τοποθετηθεί σε μια συνάρτηση theAnswer( ).

Ο κώδικας της πρώτης συνάρτησης έχει για κάθε δοκιμασία διαφορετική μορφή καθώς γράφει μαζί με την απάντηση του παίκτη, τα προαπαιτούμενα που θεωρεί η δοκιμασία δεδομένα και είναι διαφορετικά σε κάθε μια από αυτές.

Κώδικας 40: Λειτουργία δεύτερης συνάρτησης Python

```
import unittest,sys,traceback,os
from unittest.mock import patch

def main2():
    runningException = False
    try:
        import playerCode
        playerCode.theAnswer()
    except:
        runningException = True
        result = traceback.format_exc()
        return result.split('.py",')[-1]

    if not runningException:
        from playerCode import Hero

        class TestOOP(unittest.TestCase):
            @patch.object(Hero,'usePotion',autospec=True)
            def test_OOP(self, mockusePot):
                playerCode.theAnswer()
                mockusePot.assert_called()

        def suite():
            suite = unittest.TestSuite()
            suite.addTest(TestOOP('test_OOP'))
            return suite

        result = unittest.TestResult()
        suite().run(result)
        if not result.failures:
            return ("nothing")
        else:
            return (result.failures[0][1].split("Assertion")[-1])
```

Η δεύτερη συνάρτηση Python, δοκιμάζει (try) την εισαγωγή του αρχείου που δημιουργήθηκε από την πρώτη συνάρτηση. Εάν υπάρχουν σφάλματα (except) διότι ο κώδικας δεν είναι συντακτικά σωστός, αυτά επιστρέφονται για να προβληθούν στον παίκτη. Αυτό το κομμάτι είναι ίδιο για όλες τις δοκιμασίες. Στη συνέχει ελέγχεται εάν ο παίκτης δημιούργησε κώδικα που έχει τα αποτελέσματα που ορίζει η δοκιμασία. Σε περίπτωση αποτυχίας των τεστ, επιστρέφεται με κατάλληλη επεξεργασία, το πρόβλημα που υπάρχει και προβάλλεται στην οθόνη του παίκτη. Οι παραπάνω συναρτήσεις Python επιστρέφουν τιμές στη συνάρτηση C++ που τις καλεί, η οποία στη συνέχει αναλαμβάνει

να επιστρέψει τα αποτελέσματα της Python στην μηχανή παιχνιδιού και τελικά στον παίκτη. Κάθε δοκιμασία, έχει δικό της κώδικα τεστ, ο οποίος είναι διαφορετικός από τις άλλες καθώς σε κάθε δοκιμασία απαιτούνται διαφορετικά αποτελέσματα.

Οι κώδικες που παρουσιάζονται προέρχονται από τη δοκιμασία 1 του επιπέδου 5.

7 Πιλοτική αξιολόγηση παιχνιδιού

Μετά την ολοκλήρωση του παιχνιδιού, πραγματοποιήθηκε πιλοτική αξιολόγησή του από φοιτητές προπτυχιακού και μεταπτυχιακού επιπέδου σε εθελοντική βάση. Οι συμμετέχοντες, παίζοντας το παιχνίδι σχημάτισαν τις δικές τους εμπειρίες και απόψεις σχετικά με αυτό, οι οποίες έγινε προσπάθεια να καταγραφούν με χρήση ανώνυμου ερωτηματολογίου.

Σκοπός της αξιολόγησης είναι να διερευνηθεί ο βαθμός στον οποίο το παιχνίδι που αναπτύχθηκε επιτυγχάνει τους στόχους βάσει των οποίων σχεδιάστηκε. Οι στόχοι ορίστηκαν στο Κεφάλαιο 3 κατά την αρχική περιγραφή και σχεδίαση του παιχνιδιού και περιλαμβάνουν τα ζητήματα διδασκαλίας της γλώσσας Python και το επίπεδο γνώσης που έχει ένας παίκτης μετά την ενασχόλησή του με το παιχνίδι σε σχέση με το προγραμματιστικό του υπόβαθρο. Επίσης, διερευνώνται θέματα παρουσίασης και εμφάνισης του παιχνιδιού και του ψυχαγωγικού μέρους καθώς συμπεριλαμβάνονται στην σχεδίασή του. Ακόμη, εξετάζονται και τεχνικά θέματα που είναι σημαντικά για το υπό ανάλυση παιχνίδι και με συμμόρφωση στα οποία πραγματοποιήθηκε η ανάπτυξή του.

Η αξιολόγηση έγινε με χρήση ερωτηματολογίου. Το ερωτηματολόγιο που χρησιμοποιήθηκε προέρχεται από το μοντέλο αξιολόγησης MEEGA+ (Petri et al., 2016). Σύμφωνα με τους Petri et al., (2016) το μοντέλο επικεντρώνεται στην αξιολόγηση εκπαιδευτικών παιχνιδιών σε σχέση με την εμπειρία παίκτη (player experience) και τα μαθησιακά αποτελέσματα όπως αυτά γίνονται αντιληπτά από τους παίκτες (perceived learning). Επιπλέον, οι Petri et al., (2016) αναλύουν την εμπειρία παίκτη στις εξής υποκατηγορίες: εστίαση προσοχής (focused attention), διασκέδαση (fun), πρόκληση (challenge), αυτοπεποίθηση (confidence), σχετικότητα (relevance), ικανοποίηση (satisfaction), χρηστικότητα (usability) και κοινωνική αλληλεπίδραση (social interaction).

Στο ερωτηματολόγιο που δημιούργησαν οι Petri et al. (2016), βάσει του μοντέλου MEEGA+, χρησιμοποιούνται ερωτήσεις κλειστού τύπου με κλίμακα Likert. Η κλίμακα έχει διακύμανση από -2 σε περίπτωση που ο ερωτηθείς Διαφωνεί κάθετα, -1 σε περίπτωση που απλώς Διαφωνεί, 0 σε περίπτωση που Ούτε διαφωνεί ούτε συμφωνεί, 1 σε περίπτωση που Συμφωνεί και 2 σε περίπτωση που Συμφωνεί απόλυτα με την

ερώτηση. Επίσης, στο τέλος του ερωτηματολογίου υπάρχουν και κάποιες ερωτήσεις ανοιχτού τύπου.

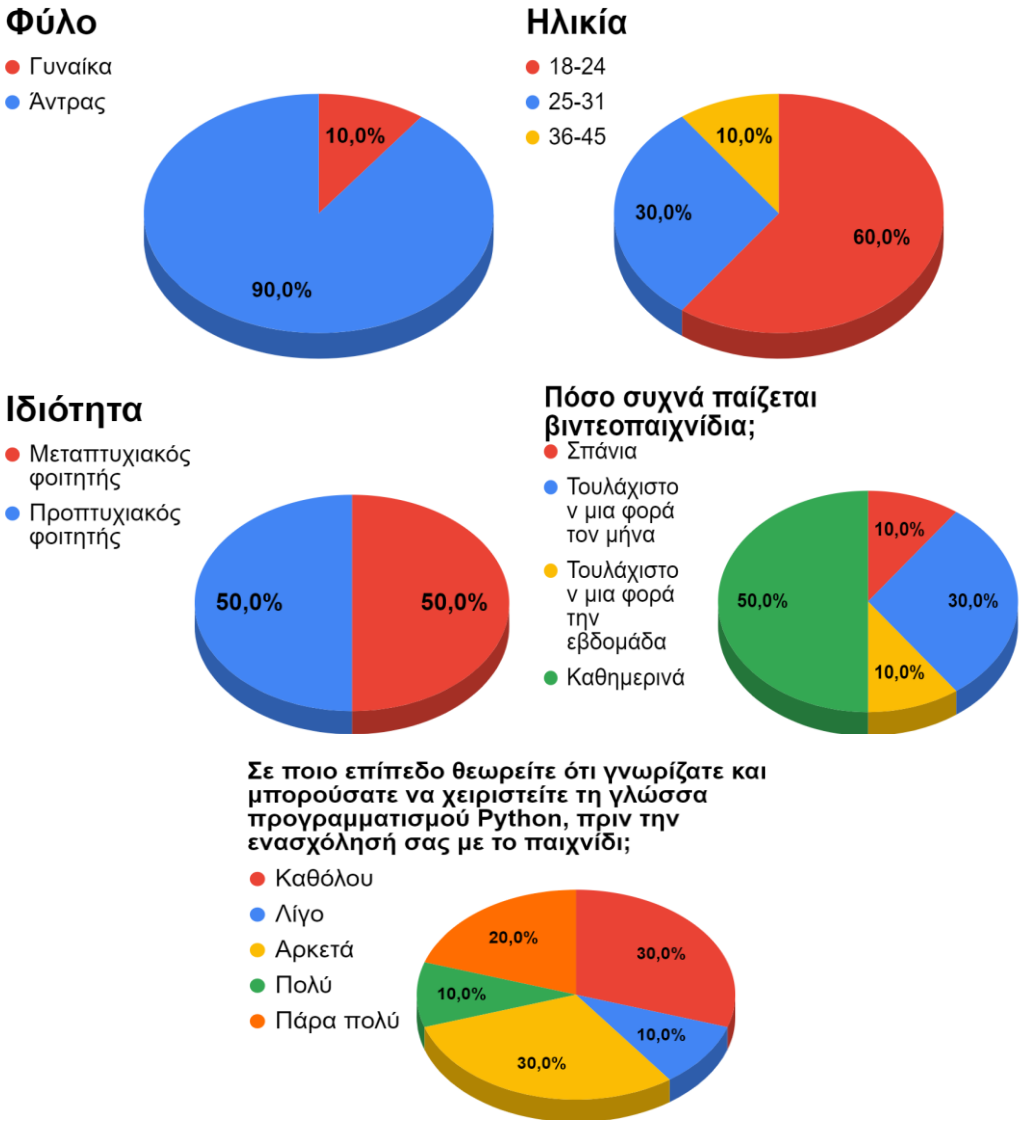
Οι ενδεικτικές ερωτήσεις που παρέχονται από τους δημιουργούς του ερωτηματολογίου, τροποποιήθηκαν για το θέμα και το αντικείμενο του παιχνιδιού που αξιολογείται. Στο ερωτηματολόγιο προστέθηκαν επίσης, ερωτήσεις που εξυπηρετούν τα ερευνητικά ερωτήματα της αξιολόγησης όπως ερωτήσεις σχετικά με τις δυνατότητες συστήματος όπου έπαιξαν το παιχνίδι οι συμμετέχοντες καθώς και αν αντιμετώπισαν προβλήματα εγκατάστασης και εκτέλεσης ή προβλήματα κατά την λειτουργία. Τέλος, αφαιρέθηκαν οι ερωτήσεις από το ερωτηματολόγιο που δεν σχετίζονται με το περιεχόμενο και τις δυνατότητες του παιχνιδιού, όπως για παράδειγμα οι ερωτήσεις που αφορούν στην κοινωνική αλληλεπίδραση που δεν υποστηρίζεται μέσω του παιχνιδιού.

7.1 Αποτελέσματα αξιολόγησης

Στην αξιολόγηση συμμετείχαν 10 φοιτητές προπτυχιακού και μεταπτυχιακού επιπέδου σπουδών του τμήματος Εφαρμοσμένης Πληροφορικής του Πανεπιστημίου Μακεδονίας . Τα αποτελέσματα από τις απαντήσεις που δόθηκαν παρουσιάζονται στις επόμενες ενότητες.

7.1.1 Δημογραφικά αποτελέσματα

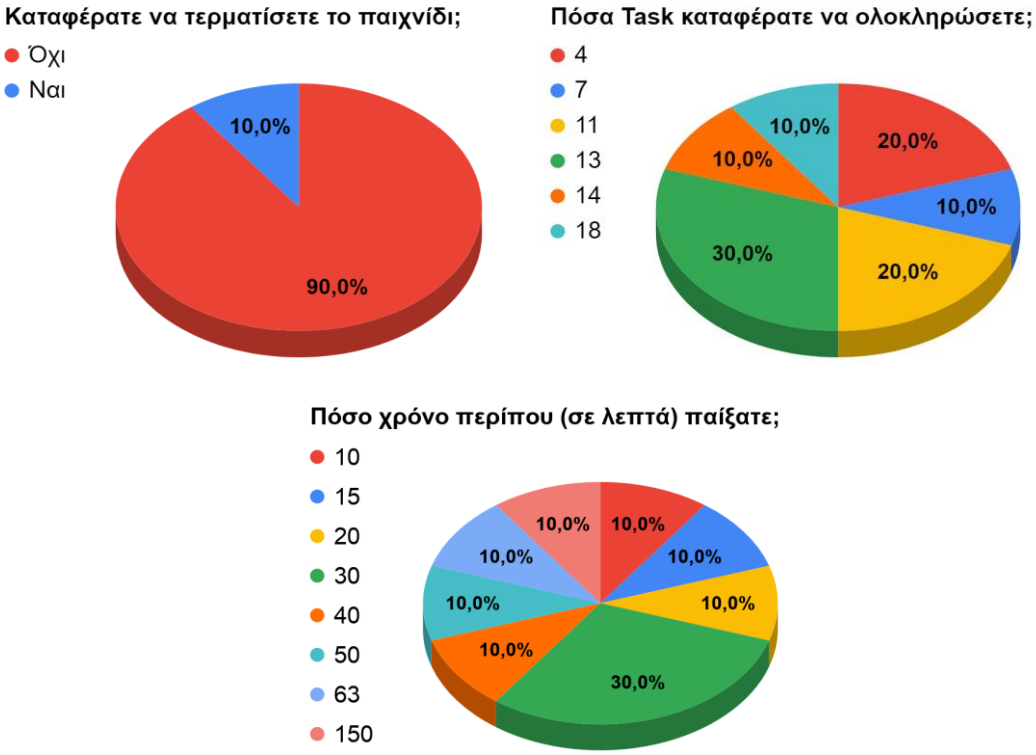
Από τα δημογραφικά αποτελέσματα φαίνεται ότι οι περισσότεροι συμμετέχοντες ήταν άντρες σε ποσοστό 90%, νεαρής ηλικίας 18 έως 24 ετών σε ποσοστό 60%, ενώ τα ποσοστά μεταπτυχιακών και προπτυχιακών φοιτητών που συμμετείχαν είναι μοιρασμένα. Επίσης, είναι σημαντικό το γεγονός ότι το μεγαλύτερο ποσοστό των συμμετεχόντων (ποσοστό 60%) παίζουν συχνά βιντεοπαιχνίδια, είτε καθημερινά είτε μία φορά την εβδομάδα, και συνεπώς έχουν μεγάλη εμπειρία σε ψυχαγωγικά παιχνίδια. Αυτό σημαίνει ότι οι απόψεις τους για το παιχνίδι βασίζονται στη γενικότερη εμπειρίας τους με τα βιντεοπαιχνίδια και τη συστηματικής ενασχόλησής τους με αυτά. Τέλος, το μεγαλύτερο μέρος των συμμετεχόντων έχει κάποια προηγούμενη εμπειρία με την γλώσσα προγραμματισμού Python, γεγονός που συμβάλει στην καλύτερη αξιολόγηση και εκτίμηση των προσφερόμενων από το παιχνίδι γνώσεων. Παρόλα αυτά, οι απόψεις των ατόμων με λίγες ή καθόλου γνώσεις στη Python είναι εξίσου σημαντικές ως σημείο αναφοράς των γνώσεων που μπορεί κάποιος που εισάγεται σε αυτή την γλώσσα να αποκτήσει από το παιχνίδι.



Εικόνα 43: Δημογραφικά αποτελέσματα

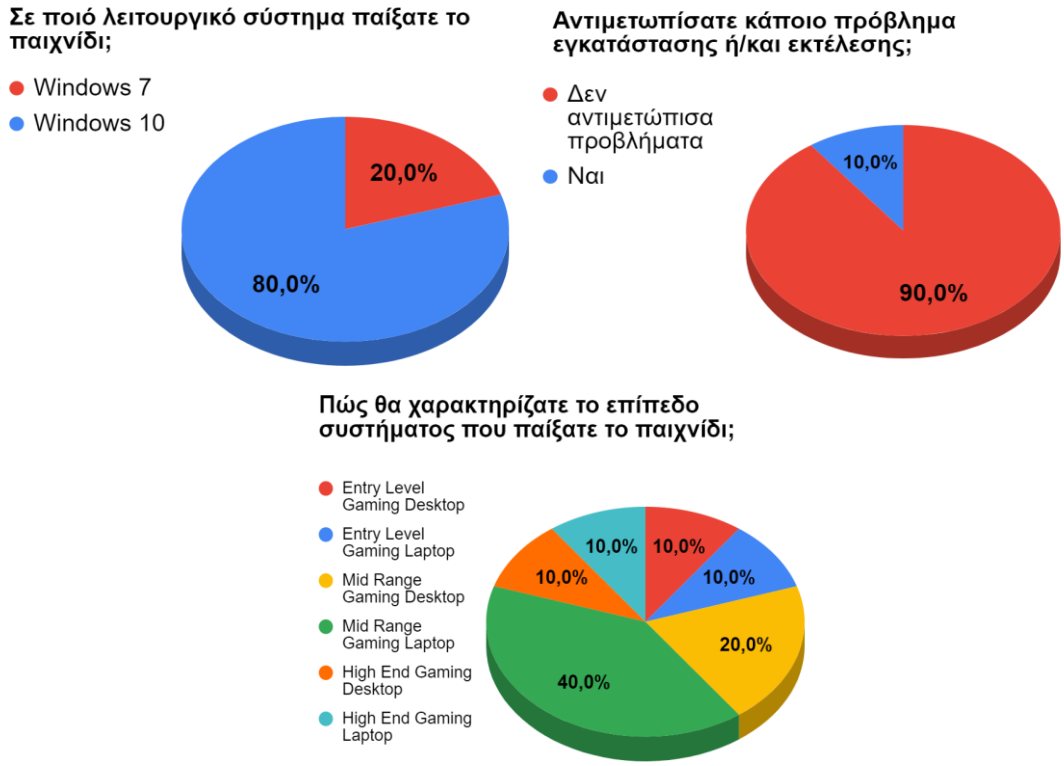
Τα αποτελέσματα σε σχέση με το παιχνίδι παρουσιάζουν ενδιαφέρον καθώς παρόλο που οι περισσότεροι συμμετέχοντες παίζουν συστηματικά βιντεοπαιχνίδια και το μεγαλύτερο ποσοστό τους έχει γνώσεις Python, δεν κατάφεραν να τερματίσουν το παιχνίδι. Αυτό, αποτυπώνεται και στα ποσοστά ολοκλήρωσης των δοκιμασιών όπου τα αποτελέσματα δεν παρουσιάζουν ομογένεια ή κάποια τάση, με τους συμμετέχοντες να ολοκληρώνουν διαφορετικό αριθμό δοκιμασιών. Τα ποσοστά των χρόνων που έπαιξαν οι παίκτες παρουσιάζουν μια ομοιογένεια ως προς το γεγονός ότι οι περισσότεροι συμμετέχοντες (70%) ασχολήθηκαν με το παιχνίδι περισσότερο από 30 λεπτά. Ίσως σε αυτό το σημείο να υπάρχει ένα πρώτο εύρημα σε σχέση με το παιχνίδι και το επίπεδο δυσκολίας του καθώς οι χρόνοι παιχνιδιού από τους συμμετέχοντες, τα ποσοστά

προϋπαρχόντων γνώσεων και ενασχόλησης με βιντεοπαιχνίδια δεν συμβαδίζουν με τα ποσοστά ολοκλήρωσης των δοκιμασιών και του παιχνιδιού συνολικά. Σε σχέση με αυτή την οπτική γίνονται πιο συγκεκριμένες ερωτήσεις στη συνέχεια του ερωτηματολογίου.



Εικόνα 44: Αποτελέσματα σχετικά με το παιχνίδι

Τέλος, στους παίκτες δεν παρουσιάστηκε κάποιο σημαντικό πρόβλημα είτε εγκατάστασης είτε εκτέλεσης είτε κατά τη λειτουργία του παιχνιδιού ασχέτως της έκδοσης του λειτουργικού συστήματος Windows που διαθέτουν και του επιπέδου δυνατοτήτων του εξοπλισμού που διαθέτουν. Το μοναδικό πρόβλημα που αναφέρθηκε από ένα συμμετέχοντα είναι το εξής: *«Υπήρξαν 2 bugs, στο 2ο επίπεδο περπατούσα στον αέρα για λίγο, και στο 6ο ένας από τους εχθρούς δεν "καταλάβαινε" ότι ήμουν μπροστά του και έτσι δεν μου επιτέθηκε»*.



Εικόνα 45: Αποτελέσματα σχετικά με το σύστημα των συμμετεχόντων

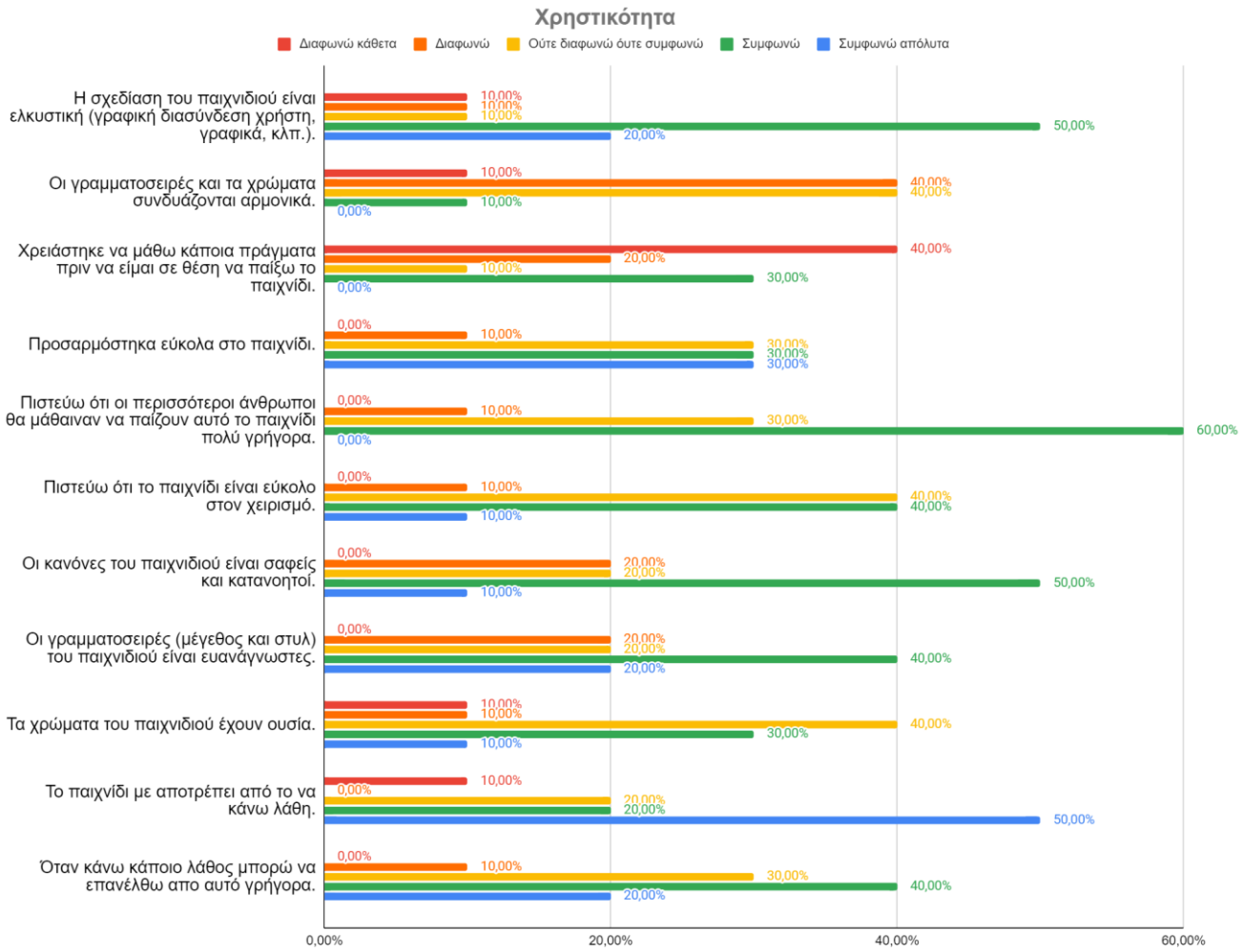
Συνολικά, το παιχνίδι κατάφερε να πετύχει τον στόχο της λειτουργίας του ανεξαρτήτως συστήματος με εγκατάσταση ή όχι της γλώσσας προγραμματισμού Python αλλά και επιπέδου δυνατοτήτων συστήματος καθώς και της έκδοσης του λειτουργικού συστήματος των Windows. Οι παίκτες, βάσει των απαντήσεων, δεν αντιμετώπισαν ιδιαίτερα προβλήματα είτε προαπαιτούμενων είτε συμβατότητας και το παιχνίδι μπορεί να θεωρηθεί ότι θα ήταν λειτουργικό σε διάφορα συστήματα με διαφορετικές δυνατότητες.

7.1.2 Εμπειρία παίκτη

Οι παίκτες αξιολόγησαν το παιχνίδι θετικά ως προς την εμπειρία που αποκόμισαν από αυτό.

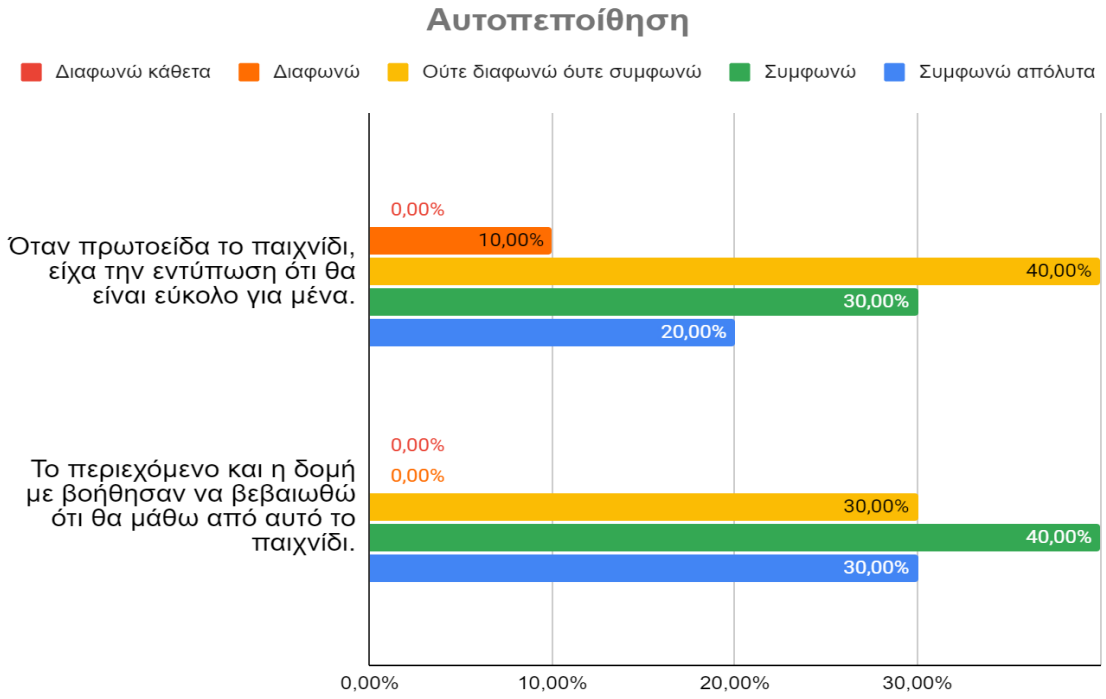
Σε σχέση με θέματα που αφορούν την *χρηστικότητα* του παιχνιδιού, η πλειοψηφία των παικτών συμφωνεί ότι *το παιχνίδι είναι εύκολο στον χειρισμό* με τους *περισσότερους ανθρώπους να είναι σε θέση να το μάθουν εύκολα* αλλά και *να προσαρμοστούν σε αυτό χωρίς να χρειαστεί να μάθουν κάτι πριν παίξουν*. Επίσης, συμφωνούν σε μεγάλο ποσοστό ότι *το παιχνίδι έχει σαφείς κανόνες*, τους *αποτρέπει από λάθη και μπορούν να επανέλθουν εύκολα από αυτά*.

Οι απόψεις όμως διίστανται σε σχέση με τις επιλεγμένες γραμματοσειρές και τα χρώματα του παιχνιδιού. Συγκεκριμένα, το 50% των φοιτητών θεωρούν ότι *οι γραμματοσειρές και τα χρώματα δεν συνδυάζονται* αρμονικά. Παρόλα αυτά, δίνουν εύσημα στα γραφικά και την σχεδίαση σε ποσοστό 70%.

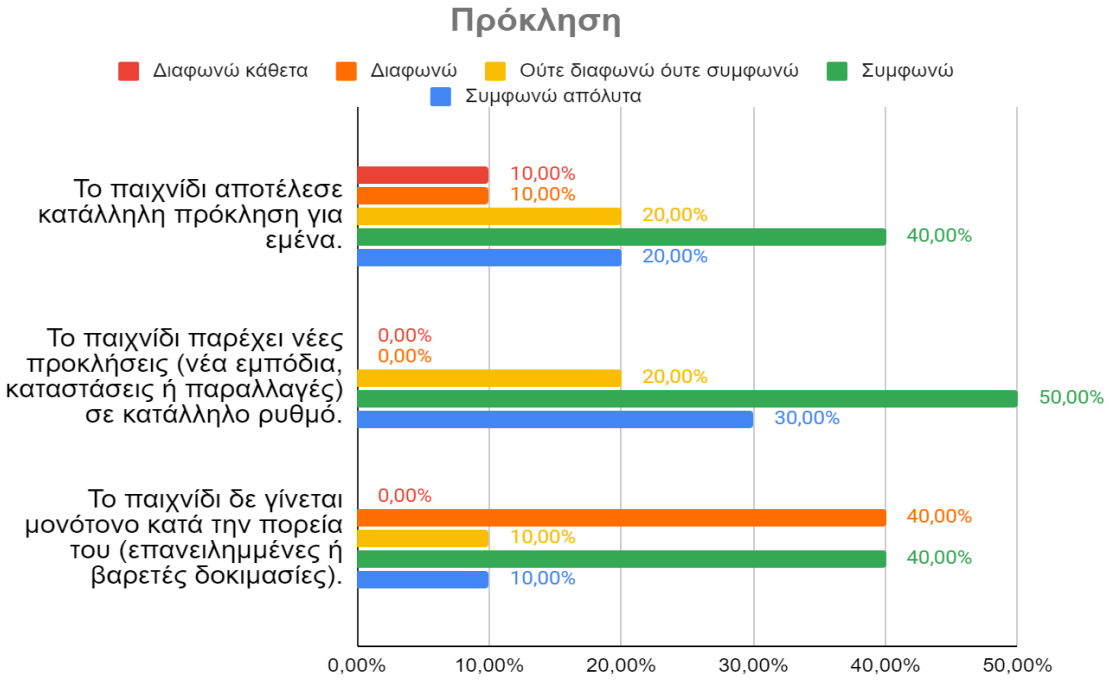


Εικόνα 46: Αποτελέσματα χρηστικότητας

Σε σχέση με την *αυτοπεποίθηση* και την *πρόκληση* που τους παρέχει το παιχνίδι, οι παίκτες έχουν γενικά θετικές απόψεις. Συγκεκριμένα, οι παίκτες θεώρησαν *το παιχνίδι θα είναι σχετικά εύκολο* όταν το είδαν πρώτη φορά και έχουν θετική άποψη σε ποσοστό 70% για το *εκπαιδευτικό περιεχόμενο και τη δομή* του. Σε θέματα *πρόκλησης*, οι συμμετέχοντες αξιολογούν θετικά το παιχνίδι με εξαίρεση το γεγονός ότι γίνεται *μονότονο κατά την πορεία του*, όπου οι απόψεις τους είναι μοιρασμένες.

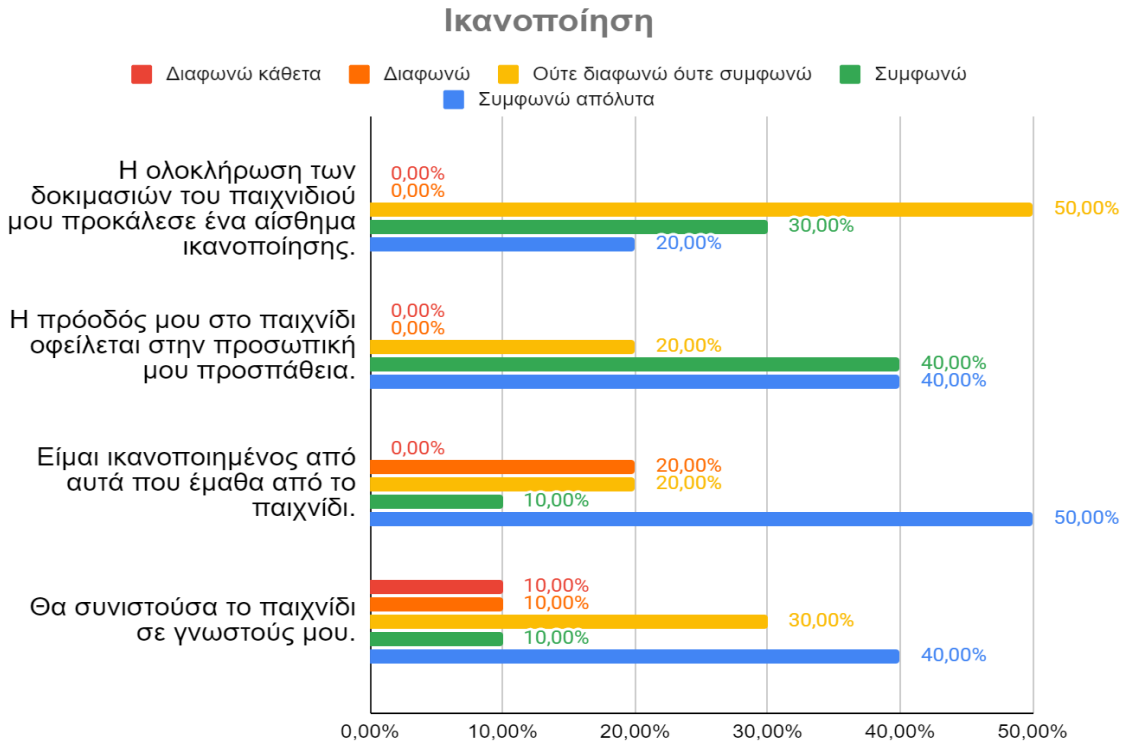


Εικόνα 47: Αποτελέσματα αυτοπεποίθησης



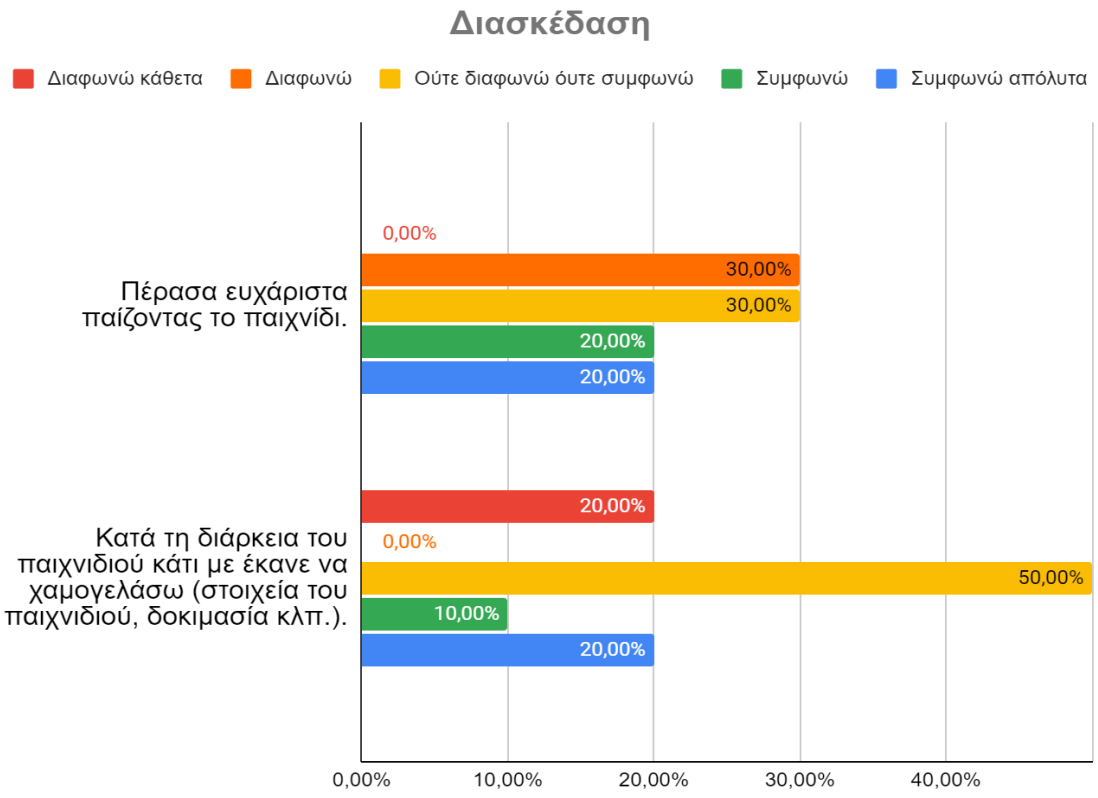
Εικόνα 48: Αποτελέσματα πρόκλησης

Σε σχέση με την *ικανοποίηση* που λαμβάνουν οι παίκτες από το παιχνίδι, οι αξιολογήσεις είναι θετικές ή τείνουν προς το θετικό σε όλες τις περιπτώσεις. Δεν υπάρχουν ιδιαίτερα μεγάλα ποσοστά αποτροπής των παικτών λόγω έλλειψης ικανοποίησης ή προόδου στο παιχνίδι.



Εικόνα 49: Αποτελέσματα ικανοποίησης

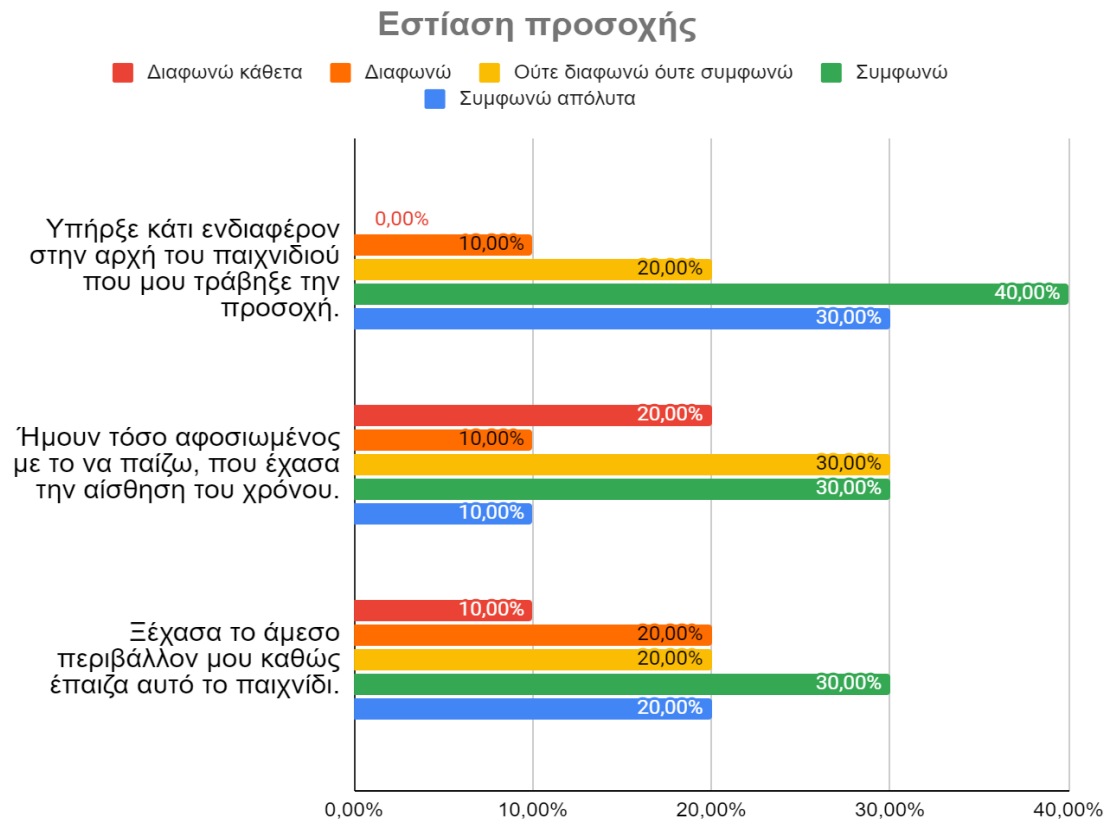
Στην κατηγορία ερωτήσεων σε σχέση με την *διασκέδαση*, οι παίκτες έχουν μοιρασμένες απόψεις σε σχέση με την ευχάριστη πάροδο του χρόνου κατά τη διάρκεια του παιχνιδιού και σε σχέση με την επαφή τους με κάποιο ιδιαίτερο χαρακτηριστικό του παιχνιδιού. Παρόλα αυτά, καθώς το παιχνίδι είναι ένα παιχνίδι σοβαρού σκοπού και περιέχει την ψυχαγωγία ως παράγοντα είναι σημαντικό το γεγονός ότι οι παίκτες δεν είχαν ιδιαίτερα αρνητικές απόψεις γι' αυτό το κομμάτι.



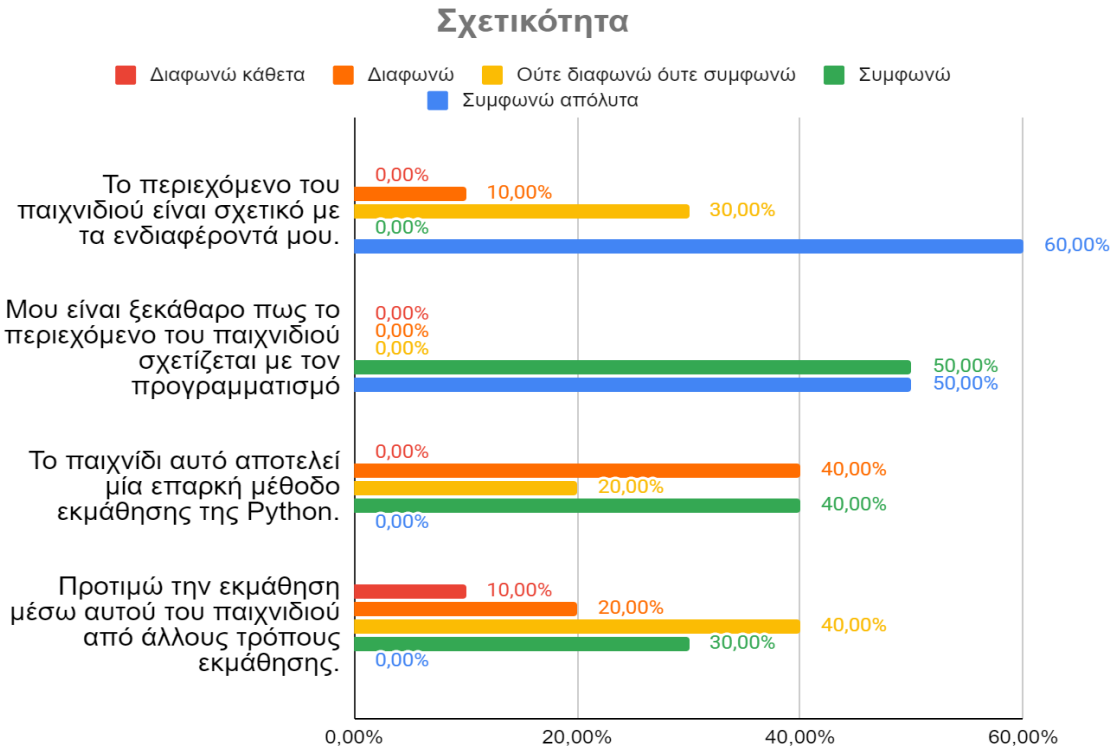
Εικόνα 50: Αποτελέσματα διασκέδασης

Σε αντίθεση όμως με τις αξιολογήσεις για τη διασκέδαση, τα ποσοστά των παικτών που βρήκαν κάτι ενδιαφέρον στο παιχνίδι είναι αρκετά μεγάλα. Ενδιαφέρον όμως παρουσιάζει το γεγονός ότι οι απόψεις τους σε σχέση με τα επίπεδα *αφοσίωσής* τους είναι μοιρασμένες και ενώ το παιχνίδι προκάλεσε το *ενδιαφέρον* τους, αυτό δεν μπόρεσε να τους κρατήσει ιδιαίτερα *προσηλωμένους*.

Όσον αφορά τη *σχετικότητα*, η πλειοψηφία των συμμετεχόντων θεωρεί ότι το περιεχόμενο του παιχνιδιού είναι σχετικό με τα ενδιαφέροντά τους, ενώ το σύνολο των συμμετεχόντων συμφωνεί ότι το εκπαιδευτικό περιεχόμενο του παιχνιδιού σχετίζεται με το αντικείμενο που πραγματεύεται. Όμως, οι απόψεις τους σε σχέση με την ικανότητα του παιχνιδιού να διδάξει τη γλώσσα προγραμματισμού Python αλλά και της προτίμησής τους για το συγκεκριμένο παιχνίδι γι' αυτό τον σκοπό διίστανται. Αυτό πιθανώς να σχετίζεται με το μεγάλο ποσοστό προϋπάρχουσας γνώσης από τους παίκτες που παρατηρήθηκε στην καταγραφή των δημογραφικών χαρακτηριστικών.



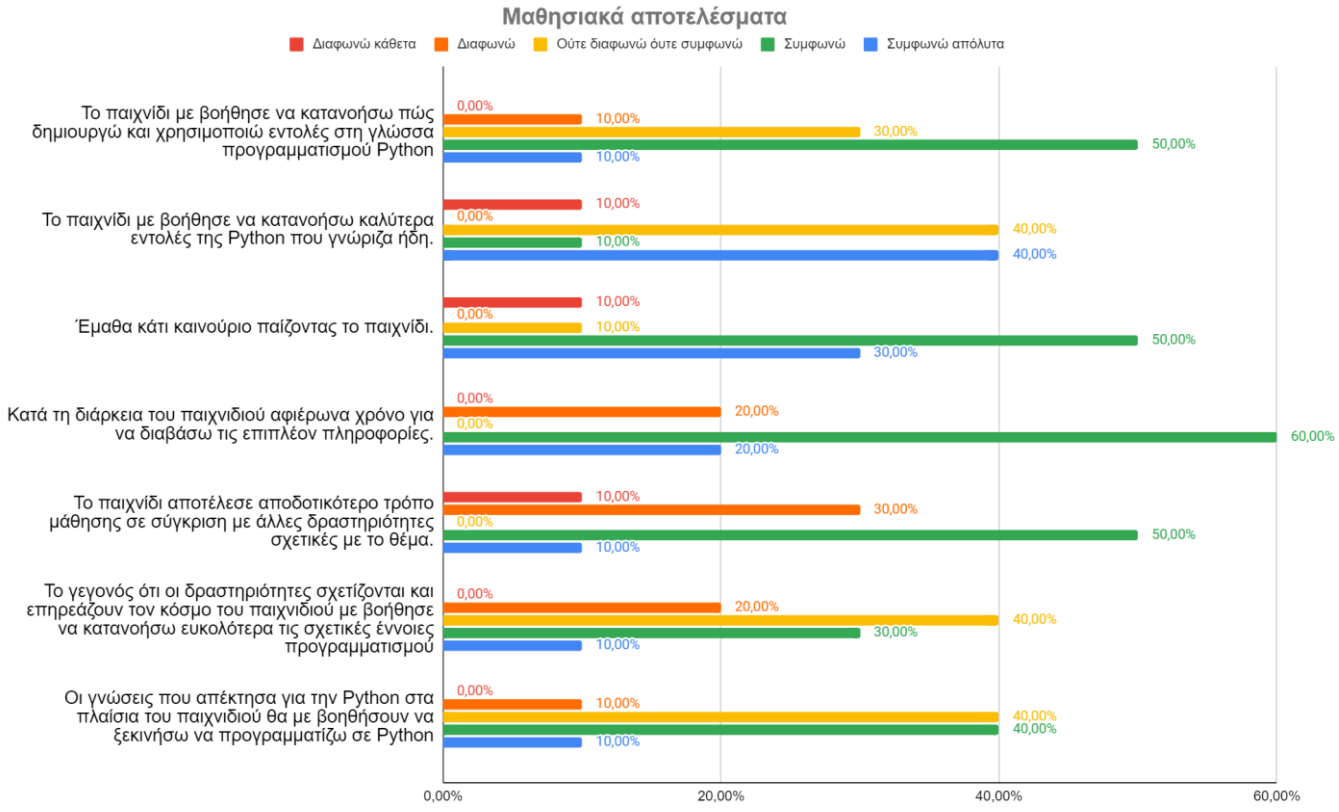
Εικόνα 51: Αποτελέσματα εστίασης προσοχής



Εικόνα 52: Αποτελέσματα σχετικότητας

7.1.3 Μαθησιακά αποτελέσματα

Τα μαθησιακά αποτελέσματα που πέτυχε το παιχνίδι, κατά την άποψη των παικτών, είναι πολύ ενθαρρυντικά. Σε μεγάλα ποσοστά οι συμμετέχοντες συμφωνούν ότι το παιχνίδι τους βοήθησε σε θέματα κατανόησης και χρήσης της γλώσσας προγραμματισμού Python. Επίσης, ενδιαφέρον παρουσιάζει το γεγονός ότι σε πολύ μεγάλο ποσοστό οι παίκτες συμφωνούν ότι έμαθαν κάτι νέο από το παιχνίδι, παρόλο που το μεγαλύτερο μέρος των συμμετεχόντων γνώριζαν ήδη Python σε αρκετά καλό επίπεδο. Αυτό αποτυπώνεται καλύτερα στο γεγονός ότι θεώρησαν το παιχνίδι καλύτερη μέθοδο εκμάθησης Python από άλλες και μπόρεσαν σε μεγάλο ποσοστό να εμπεδώσουν γνώσεις που ήδη είχαν, σε αντίθεση με τα αποτελέσματα σχετικότητας που παρουσιάστηκαν. Επίσης, με ουδέτερη προς θετική άποψη θεωρούν ότι το παιχνίδι τους παρείχε γνώσεις που θα τους βοηθήσουν να ξεκινήσουν τον προγραμματισμό με χρήση Python. Τέλος, το γεγονός ότι οι δραστηριότητες του παιχνιδιού επηρεάζουν τον κόσμο και τον χαρακτήρα απέσπασε κυρίως ουδέτερες αξιολογήσεις με τις θετικές μόλις που ξεπερνούν τις αρνητικές και τους παίκτες να μην εμφανίζουν σημαντική προτίμηση σε αυτή τη σχεδιαστική επιλογή.



Εικόνα 53: Μαθησιακά αποτελέσματα

7.1.4 Ερωτήσεις ανοικτού τύπου

Εκτός από τις ερωτήσεις κλειστού τύπου το ερωτηματολόγιο περιελάμβανε και τρεις ερωτήσεις ανοικτού τύπου προκειμένου οι συμμετέχοντες να εκφράσουν ελεύθερα τις απόψεις τους για το παιχνίδι. Σε σχέση με τα *θετικά σημεία του παιχνιδιού* οι απαντήσεις που δόθηκαν σχετίζονται με τα υψηλού επιπέδου γραφικά που έχει το παιχνίδι αλλά και τον σαφή χειρισμό του. Ενδεικτικές απαντήσεις:

- *«Μου άρεσε το είδος του παιχνιδιού, τα γραφικά και αρκετά η μουσική. Αρκετά ενδιαφέρον.»*
- *«τα γραφικά και ο χειρισμός»*
- *«Αξίζει να αναφερθεί ότι προσωπικά το βρήκα πρωτότυπο και το θέμα μου άρεσε πάρα πολύ (προγραμματισμό και κάστρο με πολεμιστές κλπ). Θεωρώ επίσης ότι λόγω της Unreal Engine που χρησιμοποιήθηκε τα γραφικά ξεκινάν ήδη από ένα καλό επίπεδο και μπορούν να φτάσουν ακόμα και σε επίπεδα 3Α. Εντυπωσιάστηκα όταν είδα πως εκτός των απλών rpg χαρακτηριστικών (επίθεση, άμυνα, stats), μπορούσα να χρησιμοποιήσω και skills. Όσον αφορά το εκπαιδευτικό κομμάτι, η κλιμάκωση της δυσκολίας είναι καλή και τα errors είναι αρκετά βοηθητικά.»*

Σε σχέση με την άποψή τους σε *θέματα βελτίωσης του παιχνιδιού*, οι παίκτες προτείνουν την καλύτερη παρουσίαση του εκπαιδευτικού περιεχομένου αλλά και των γραμματοσειρών που χρησιμοποιούνται στο παιχνίδι. Ενδεικτικές απαντήσεις:

- *«Θα μπορούσε να γίνει λίγο πιο ελκυστικό το κομμάτι της θεωρίας. Ειδικά τα γράμματα είναι μικρά και πιστεύω σε συνδυασμό με το υπόλοιπο παιχνίδι αυτό το κομμάτι υστερεί. Επίσης, οι εχθροί σε κάποιες περιπτώσεις δεν καταλάβαιναν ότι ο ήρωας βρισκόταν δίπλα τους.»*
- *«Κατά την γνώμη μου το Ui (menu/in game) έχει μεγάλα περιθώρια βελτίωσης τόσο στα χρώματα όσο και γενικότερα αισθητικά (ίσως και επιπλέον animations). Βρήκα πολύ βοηθητικές τις έτοιμες λύσεις και θεωρώ πως θα ήταν καλό ο παίκτης να ξεκινά με λιγότερες και όσο προοδεύει να επιβραβεύεται, ώστε να μην κάνει κατάχρηση αυτών αλλά να μπορεί να δεχτεί βοήθεια. Τα controls του παίκτη είναι πολύ καλά και θα μπορούσαν να αναδειχθούν ακόμα περισσότερα με κάποιο λίγο πιο*

λεπτομερές animation (ομοίως και για τα npcs). Επιπλέον, θα μπορούσε να υπάρχει και story mode που θα παίζεις το παιχνίδι χωρίς μόνο για την ιστορία, χωρίς τις ερωτήσεις προγραμματισμού, όμως για να μην παραμερίζεται το εκπαιδευτικό κομμάτι ίσως να "ξεκλειδωνόταν" αφού πρώτα τερματιστεί η βασική έκδοση του παιχνιδιού. Θεωρώ επίσης πως τα κείμενα θα έπρεπε να συμπτυχθούν ώστε να μην υπάρχει μεγάλη κάμψη στον ρυθμό του παιχνιδιού. Προαιρετικά θα μπορούσαν να γίνει προσθήκη σημείων όπου θα δύνεται κάποιο κομμάτι της ιστορίας του παίκτη μας (είτε με μικρά κείμενα, είτε με κάποιο μικρό voice acting).»

Τέλος, οι παίκτες κατέγραψαν τις απόψεις τους σε σχέση με το πεδίο *και τις δυνατότητες εφαρμογής και χρήσης του παιχνιδιού*, με τις περισσότερες απόψεις να συμφωνούν στο γεγονός ότι το παιχνίδι αποτελεί ένα καλό πλαίσιο εισαγωγής στη Python αλλά και τον προγραμματισμό γενικότερα καθώς επίσης και την πιθανή αξιοποίηση του παιχνιδιού ακόμα και σε μικρότερες ηλικίες. Ενδεικτικές απαντήσεις αποτελούν:

- *«Σίγουρα θα πρέπει να προωθηθεί ως καλό εκπαιδευτικό εργαλείο. Μπορείς εύκολα να κατανοήσεις της βασικές έννοιες του προγραμματισμού καθώς και την φιλοσοφία και την μαγεία της Python !»*
- *«Θεωρώ πως είναι εύκολο ένα τέτοιο παιχνίδι να επιτελέσει τον σκοπό του εάν χρησιμοποιηθεί σε παιδιά ηλικίας 13-17 ετών. Ωστόσο, σε κάτι τέτοιο δεν πιστεύω πως υπάρχει άνω όριο, δηλαδή θα μπορούσε να παιχτεί και συνειδητοποιημένα από ενήλικες και ενδεχομένως να έχει και το θεμιτό αποτέλεσμα. Λαμβάνοντας υπόψη αυτό, θεωρώ πως 1) η διάθεση του σε χώρους όπου τα παιδιά μαθαίνουν προγραμματισμό (όπως κάποιο ειδικό φροντιστήριο -αν υπάρχει-) , 2) η διάθεση του σε ημερίδες κατά τις οποίες μαθητές επισκέπτονται τους χώρους του πανεπιστημίου και 3) η διάθεσή του σε πρωτοετείς φοιτητές (ακόμα και ως εξέταση σε κάποιο μάθημα - φυσικά σε πιο αυστηρά πλαίσια-), θα ήταν κατάλληλες περιπτώσεις για την αξιοποίηση του παιχνιδιού αυτού.»*

7.2 Συμπεράσματα

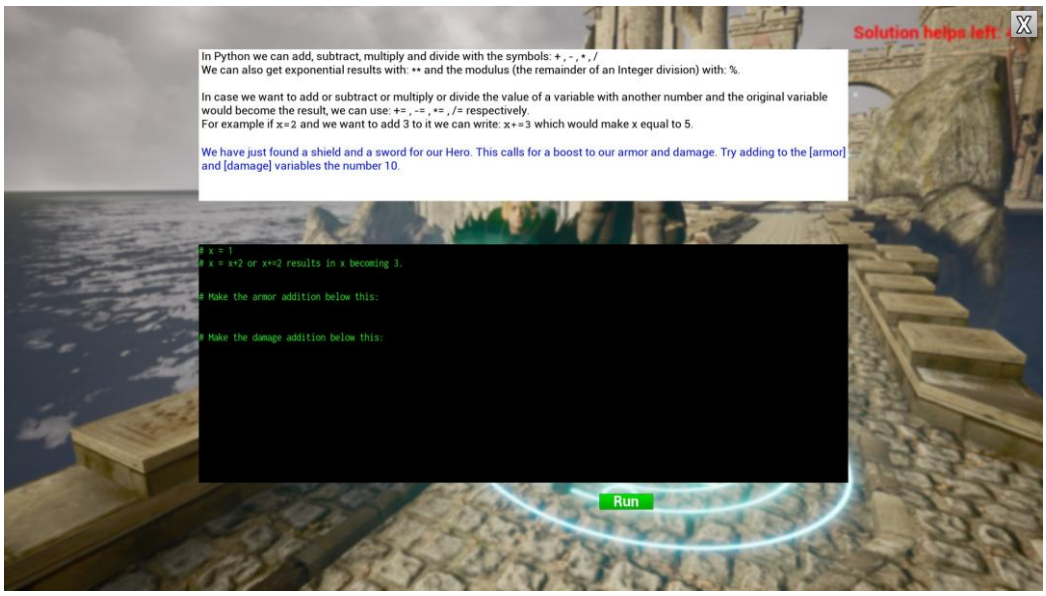
Συνολικά, το παιχνίδι κρίνεται ότι πέτυχε τους στόχους για τους οποίους δημιουργήθηκε. Σε σχέση με θέματα ψυχαγωγικού περιεχομένου οι αξιολογήσεις ήταν

θετικές τόσο σε σχέση με τα γραφικά όσο και με το gameplay. Επίσης, θετικές ήταν οι απόψεις των παικτών και στα θέματα διδασκαλίας τόσο σε σχέση με το εκπαιδευτικό περιεχόμενο της εκμάθησης της γλώσσας προγραμματισμού Python όσο και με το επίπεδο δυσκολίας του. Τέλος, δεν κατεγράφησαν υπέρμετρα αρνητικές απόψεις σε κάποια πτυχή του παιχνιδιού και οι συμμετέχοντες που δεν γνώριζαν Python κατάφεραν να παίξουν και να μάθουν από το παιχνίδι.

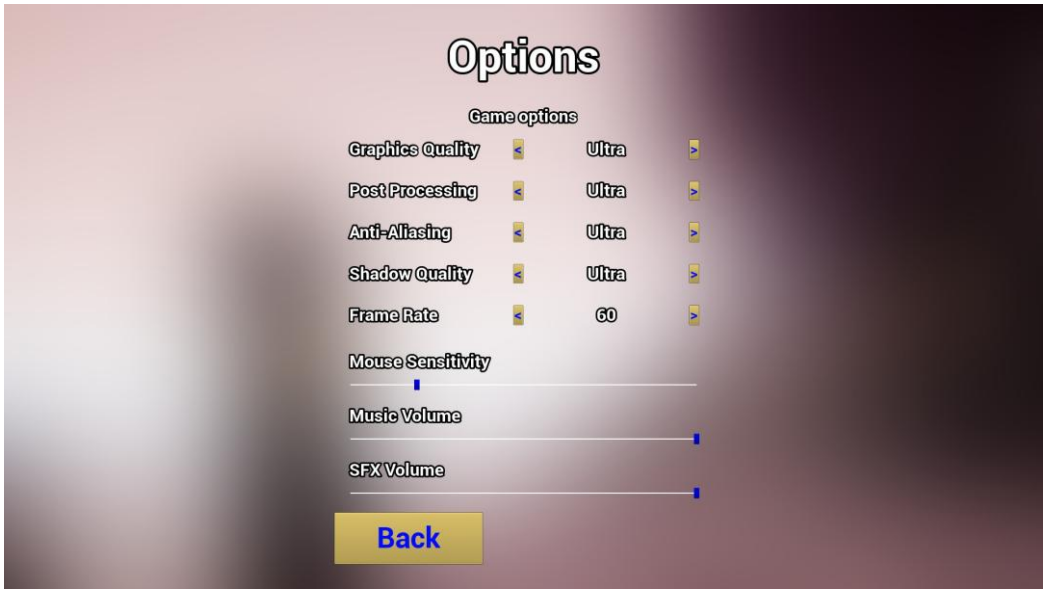
Παρόλα αυτά, υπάρχουν περιθώρια βελτίωσης τόσο σε θέματα τρόπου παρουσίασης της θεωρίας όσο και σε θέματα user interface που κυρίως παρατήρησαν οι συμμετέχοντες στην αξιολόγηση. Οι παίκτες πρότειναν διάφορους τρόπους βελτίωσης και με τις αξιολογήσεις τους βοήθησαν να δημιουργηθεί η κατεύθυνση που πρέπει να πάρει η βελτίωση του παιχνιδιού. Συγκεκριμένα, η παρουσίαση του εκπαιδευτικού περιεχομένου προς τους παίκτες, έχει άμεσο τρόπο βελτίωσης με μεγαλύτερες γραμματοσειρές και πιο ευδιάκριτα χρώματα. Επίσης, παρόμοια λογική πρέπει να ακολουθήσει και η αλλαγή σε σχέση με το user interface του παιχνιδιού στα μενού που υπάρχουν. Τέλος, θα μπορούσε το παιχνίδι να βελτιώσει περεταίρω το ψυχαγωγικό κομμάτι και να μειώσει τη μονοτονία του, σε σχέση με το εκπαιδευτικό μέρος, καθώς στην αξιολόγηση της διασκέδασης οι απόψεις ήταν ποικίλες.

7.2.1 Βελτιώσεις

Βάσει των αποτελεσμάτων του ερωτηματολογίου σε σχέση με τις απόψεις των παικτών για το παιχνίδι, πραγματοποιήθηκαν ορισμένες βελτιώσεις. Αρχικά, σε όλες τις δοκιμασίες βελτιώθηκε η παρουσίαση του εκπαιδευτικού περιεχομένου με μεγέθυνση των γραμματοσειρών και επιλογή διαφορετικών χρωμάτων για τις εκφωνήσεις των προβλημάτων. Επίσης, ο παίκτης σε κάθε σελίδα θεωρίας μπορεί να βλέπει το εκπαιδευτικό περιεχόμενο χωρίς να είναι απαραίτητη η χρήση της ροδέλας στο ποντίκι (scrolling). Ακόμη, κατά την διάρκεια κάθε δοκιμασίας, το παιχνίδι βρίσκεται σε κατάσταση “Paused” αλλά ο παίκτης είναι σε θέση να διακρίνει το περιβάλλον και δεν είναι απομονωμένος από τον κόσμο του παιχνιδιού. Τέλος, τα μενού βελτιώθηκαν σε σχέση με την ορατότητα του περιεχομένου τους. Οι αλλαγές αυτές, παρουσιάζονται στις εικόνες που ακολουθούν.



Εικόνα 54: Βελτιωμένη δοκιμασία 1 - επίπεδο 2



Εικόνα 55: Βελτιωμένο μενού "Options"

8 Επίλογος

8.1 Σύνοψη και συμπεράσματα

Η διπλωματική αυτή εργασία είχε ως σκοπό τη σχεδίαση και την ανάπτυξη ενός παιχνιδιού σοβαρού σκοπού που διδάσκει τη γλώσσα προγραμματισμού Python. Οι στόχοι της, συμπεριέλαβαν και την εισαγωγή νέων παικτών χωρίς προγραμματιστικό υπόβαθρο ή γνώσεις με τη συγκεκριμένη γλώσσα καθώς και παίκτες με μεγαλύτερη εμπειρία. Στα πλαίσια αυτά, παρουσιάστηκαν θέματα παιχνιδιών σοβαρού σκοπού και πιο συγκεκριμένα για τη διδασκαλία της Python.

Κατά τη σχεδίαση έγινε προσπάθεια ένταξης του εκπαιδευτικού περιεχομένου στο ψυχαγωγικό μέρος του παιχνιδιού καθώς και της παροχής οπτικού ερεθίσματος στους παίκτες που σχετίζεται με το εκπαιδευτικό αντικείμενο ως αποτέλεσμα της επιτυχούς επίλυσης δοκιμασιών. Επίσης, ο κώδικας Python που δημιουργείται από τους παίκτες κατά τη διάρκεια του παιχνιδιού στις επιμέρους δοκιμασίες και η προστασία τους σε συνδυασμό με την καθοδήγησή τους, ιδιαίτερα για παίκτες που δεν έχουν καμία προηγούμενη γνώση, ήταν σημαντικό μέρος της σχεδίασής του.

Όσον αφορά την υλοποίηση του παιχνιδιού, επιλέχθηκαν τεχνολογίες που θα επιτρέπουν την ενσωμάτωση της γλώσσας προγραμματισμού Python σε υψηλής ποιότητας γραφικών παιχνίδι και ικανοποιητικού gameplay. Έτσι, έγινε χρήση της μηχανής παιχνιδιών Unreal Engine 4 που έχει τη δυνατότητα να συνδυαστούν τα παραπάνω.

Τέλος, το παιχνίδι αξιολογήθηκε ως προς την ικανοποίηση των απαιτήσεων που αναλύθηκαν και τέθηκαν πριν και κατά την δημιουργία του. Οι αξιολογήσεις από τους συμμετέχοντες καθοδηγούν την πορεία βελτίωσης και πιθανής επέκτασης του παιχνιδιού.

8.2 Όρια και περιορισμοί της έρευνας

Κατά τη διάρκεια του παιχνιδιού, γίνεται προσπάθεια εισαγωγής των παικτών στη γλώσσας προγραμματισμού Python και μέσα από τις δραστηριότητες των δοκιμασιών οδηγούνται σε πιο προχωρημένες εντολές και δομές της γλώσσας. Όμως, το εύρος της λειτουργίας και των δυνατοτήτων μιας γλώσσας προγραμματισμού θα ήταν αδύνατο να καλυφθεί στα πλαίσια ενός παιχνιδιού σοβαρού σκοπού.

Επίσης, παρόλο που δημιουργήθηκε ένα σύστημα λειτουργίας παιχνιδιού (gameplay) στο οποίο εντάσσονται όλα τα στοιχεία του παιχνιδιού, ήταν αναγκαίο να εξεταστεί ο τρόπος εισαγωγής κάθε νέου στοιχείου και ο τρόπος που αλληλεπιδρά με τον παίκτη και τον κόσμο του παιχνιδιού. Το παραπάνω, σε συνδυασμό με την μη ύπαρξη ανθρωπίνων πόρων καταμερισμού υλοποίησης έργων οδηγεί σε περιορισμούς χρόνου. Έτσι, το μέγεθος του παιχνιδιού τόσο σε εκπαιδευτικό περιεχόμενο, όπως αναφέρθηκε, αλλά και σε ψυχαγωγικό περιορίζεται από την ανάπτυξή του από 1 άτομο και όχι από ομάδα προγραμματιστών.

8.3 Μελλοντικές επεκτάσεις

Καθώς παρουσιάστηκαν συγκεκριμένα θέματα κατά την αξιολόγηση του παιχνιδιού από τους συμμετέχοντες, θα ήταν λογικό ως πρώτη μελλοντική εργασία η βελτίωση των σημείων που έκριναν ως αδυναμίες ή αστοχίες του παιχνιδιού. Οι παίκτες είχαν πολύ συγκεκριμένες απόψεις για κάποια χαρακτηριστικά του παιχνιδιού και θα ήταν ένα εύκολο βήμα να γίνουν οι σχετικές βελτιώσεις προκειμένου το εκπαιδευτικό μέρος του παιχνιδιού να έχει μεγαλύτερη συνοχή με το εκπαιδευτικό του μέρος και να είναι πιο ευχάριστο για τους παίκτες. Επίσης, η βελτίωση της διασκέδασης που προσφέρει το παιχνίδι θα ήταν ένα ακόμα σημείο μελλοντικών αλλαγών αλλά ο τρόπος που θα μπορούσε να επιτευχθεί αυτό δεν είναι ξεκάθαρος και χρήζει περεταίρω μελέτης και ανάλυσης καθώς θα πρέπει να συνδυάζεται και με το εκπαιδευτικό κομμάτι του παιχνιδιού.

Υπάρχει επίσης δυνατότητα μελλοντικής επέκτασης του παιχνιδιού που αναπτύχθηκε ώστε να καλύπτει ακόμα μεγαλύτερο μέρος και υλικό της Python που προσπαθεί να διδάξει. Μαζί με το εκπαιδευτικό περιεχόμενο, θα μπορούσαν να συμπεριλαμβάνονται και γραφικά στοιχεία επέκτασης των ήδη υπαρχόντων όπως νέες κινήσεις και ικανότητες για τον ήρωα, νέες λειτουργίες και ικανότητες για τους εχθρούς ή ακόμα και νέοι κόσμοι με διαφορετικά περιβάλλοντα και νέοι ήρωες και εχθροί. Τα παραπάνω, θα πρέπει να μελετηθούν ως προς τον τρόπο ένταξής τους και να συνδεθούν με τη φιλοσοφία που υπάρχει στο παιχνίδι στο θέμα σύνδεσης αποτελέσματος δοκιμασιών με το περιβάλλον και τον παίκτη.

Επίσης, το παιχνίδι υποστηρίζει στην τελική του μορφή μόνο έναν παίκτη. Θα ήταν λογικό σε μελλοντική επέκταση, να ενσωματωθεί μαζί με τους νέους ήρωες και λειτουργία για δύο ή και παραπάνω παίκτες. Για να συμβεί αυτό, θα πρέπει να

ανασχεδιαστεί το υπάρχον σύστημα λειτουργίας από την αρχή ώστε να είναι έτοιμο για την είσοδο παραπάνω του ενός παικτών. Εάν υλοποιηθεί ένα τέτοιο σύστημα, τότε μπορεί να γίνει χρήση και του διαδικτύου για παιχνίδι από διαφορετικούς υπολογιστές που θα ήταν ευκολότερο στην υλοποίηση από ότι φαντάζεται κανείς λόγω της υποκείμενης μηχανής παιχνιδιών και των λειτουργιών που παρέχει.

Τέλος, το παιχνίδι που δημιουργήθηκε ως αποτέλεσμα της διπλωματικής εργασίας δεν περιέχει στοιχεία και λειτουργίες που θα του επέτρεπαν να ενταχθεί εύκολα στα πλαίσια της εκπαιδευτικής διαδικασίας. Σε μελλοντική επέκταση του παιχνιδιού, θα ήταν δυνατή η δημιουργία συστημάτων για διδάσκοντες ώστε να μπορούν να παρακολουθούν και να καταγράφουν την πορεία των μαθητών στο παιχνίδι. Παρόλο που μπορεί κάποιος να δει την πρόοδο ενός παίκτη μέσα από τα μενού επιλογής κόσμων και δοκιμασιών που έχει ολοκληρώσει ο παίκτης, αυτά δεν αναπτύχθηκαν με αυτό τον σκοπό. Στην παρούσα μορφή, δεν υπάρχει κάποια λειτουργία που να καταγράφει τα λάθη που έκανε ένας παίκτης κατά τη διάρκεια του παιχνιδιού ώστε να μελετηθούν και να αναλυθούν είτε από τον ίδιο είτε με τη βοήθεια κάποιου διδάσκοντα. Το πρώτο βήμα σε μια τέτοια υλοποίηση θα ήταν η δημιουργία ενός συστήματος αποθήκευσης των λαθών αλλά και των επιτευγμάτων του παίκτη, το οποίο θα δημιουργούσε προοπτικές για χρήση του παιχνιδιού που δεν συμπεριλήφθηκαν στην αρχική σχεδίαση.

Βιβλιογραφία

Checkio.org. n.d. Teaching with CheckiO ClassRooms. [online] Available at: <https://checkio.org/teachers/> [Accessed 20 January 2021]

CodeCombat. n.d. CodeCombat - Coding games to learn Python and JavaScript. [online] Available at: <https://codecombat.com/teachers/resources/faq> [Accessed 20 January 2021]

CodinGame. n.d. Coding Games and Programming Challenges to Code Better. [online] Available at: <https://www.codingame.com/faq> [Accessed 20 January 2021]

de Freitas, S. and Jarvis, S. (2006). A Framework for developing serious games to meet learner needs. Interservice/Industry Training, Simulation & Education Conference, I/ITSEC. 2006

de Freitas, S. and Oliver, M., 2006. How can exploratory learning with games and simulations within the curriculum be most effectively evaluated?. Computers & Education, 46(3), pp.249-264

Djaouti, Damien & Alvarez, Julian & Jessel, Jean-Pierre. (2011). Classifying Serious Games: the G/P/S model. Handbook of Research on Improving Learning and Motivation through Educational Games: Multidisciplinary Approaches

Docs.python.org. n.d. 1. Embedding Python in Another Application — Python 3.9.1 documentation. [online] Available at: <https://docs.python.org/3/extending/embedding.html> [Accessed 20 January 2021]

Docs.unrealengine.com. n.d. Blueprint Function Libraries. [online] Available at: <https://docs.unrealengine.com/en-US/ProgrammingAndScripting/ProgrammingWithCPP/BlueprintFunctionLibraries/index.html> [Accessed 20 January 2021]

Docs.unrealengine.com. n.d. Tools and Editors. [online] Available at: <https://docs.unrealengine.com/en-US/Basics/ToolsAndEditors/index.html> [Accessed 20 January 2021]

Fernández-Manjón, B., Moreno-Ger, P., Martinez-Ortiz, I. and Freire, M., 2015. Challenges of serious games. EAI Endorsed Transactions on Game-Based Learning, 2(6)

Gomes, A. & Mendes, A. (2007). Learning to program - difficulties and solutions. In: International Conference on Engineering Education – ICEE. (pp. 283–287)

Halpern, J., 2018. The What and Why of Game Engines. [online] Medium. Available at: <https://medium.com/@jaredehalpern/the-what-and-why-of-game-engines-f2b89a46d01f> [Accessed 20 January 2021]

Jain, A., 2020. GAME ENGINES: DESIGNING GAMES EFFICIENTLY. [online] Medium. Available at: <https://medium.com/wharf-street-studios/game-engines-designing-games-efficiently-afa87af3d0f4> [Accessed 20 January 2021]

Lameras, P., Arnab, S., Dunwell, I., Stewart, C., Clarke, S. and Petridis, P., 2017. Essential features of serious games design in higher education: Linking learning attributes to game mechanics. British Journal of Educational Technology, 48(4), pp.972-994.

Michael, D. and Chen, S., 2006. Serious games: Games That Educate Train and Inform. Boston, Mass.: Thomson Course Technology

Petri, Giani & Gresse von Wangenheim, Christiane & Borgatto, Adriano. (2016). MEEGA+: An Evolution of a Model for the Evaluation of Educational Games. Brazilian Institute for Digital Convergence

Serious.gameclassification.com. n.d. Serious Game Classification : The online classification of Serious Games. [online] Available at: <http://serious.gameclassification.com/> [Accessed 20 January 2021]

Sideris, G. and Xinogalos, S., 2019. PY-RATE ADVENTURES: A 2D Platform Serious Game for Learning the Basic Concepts of Programming With Python. Simulation & Gaming, 50(6), pp.754-770

Sousa, C. and Costa, C., 2018. Videogames as a learning tool: is game-based learning more effective?. Revista Lusófona de Educação, (40), pp.199-240.

Unreal Engine. n.d. Unreal Engine | The most powerful real-time 3D creation platform. [online] Available at: <https://www.unrealengine.com> [Accessed 20 January 2021]

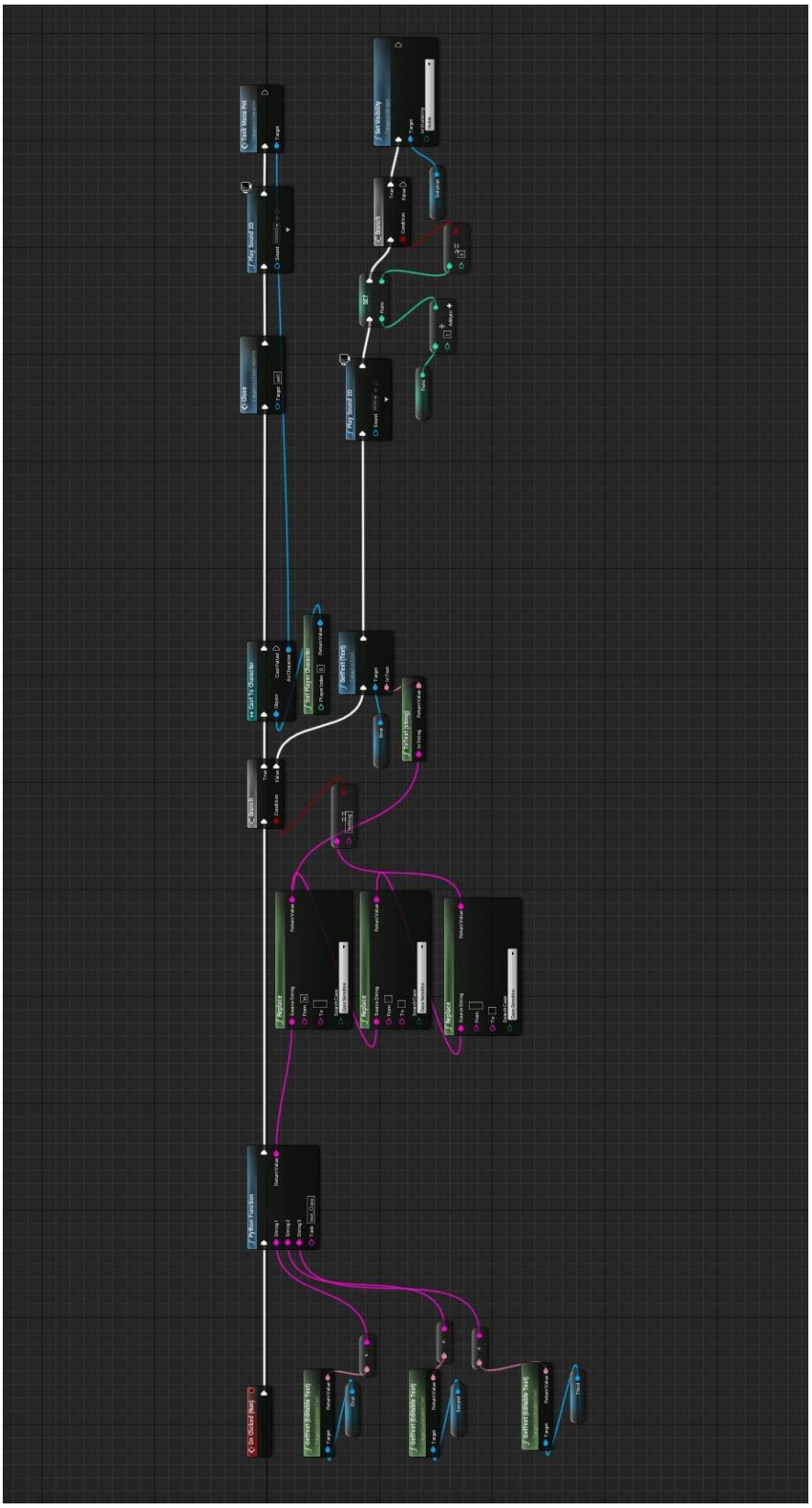
Unreal Engine. n.d. Unreal Engine | Features. [online] Available at: <https://www.unrealengine.com/en-US/features> [Accessed 20 January 2021]

Wilson, K., Bedwell, W., Lazzara, E., Salas, E., Burke, C., Estock, J., Orvis, K. and Conkey, C., 2008. Relationships Between Game Attributes and Learning Outcomes. Simulation & Gaming, 40(2), pp.217-266

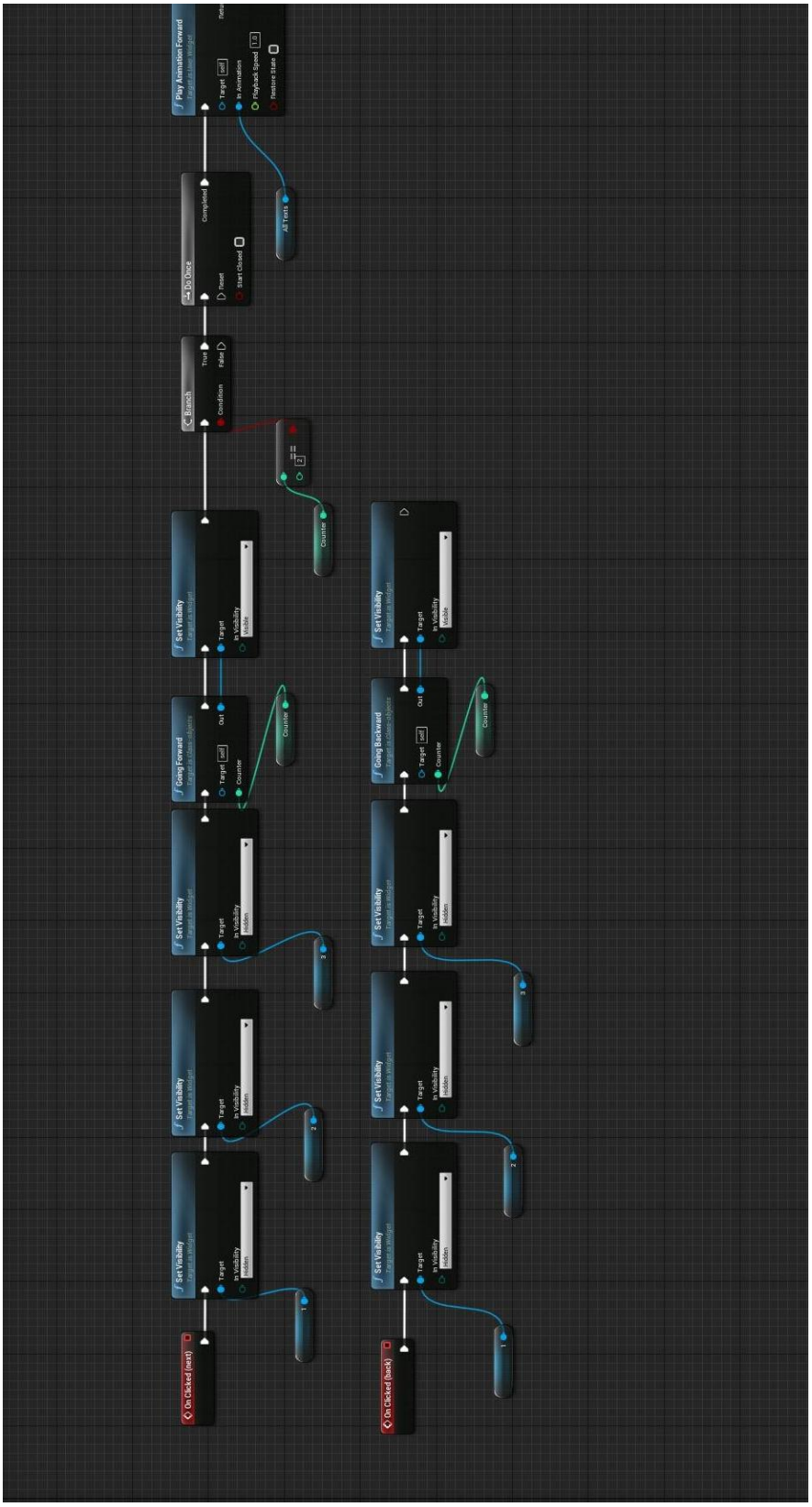
Zapušek, M. and Rugelj, J., 2013. Learning programming with serious games. EAI Endorsed Transactions on Game-Based Learning, 1(1), p.e6

Γρηγόριος Σιδέρης (2019). Ανάπτυξη παιχνιδιού σοβαρού σκοπού για την εκμάθηση εννοιών προγραμματισμού με τη γλώσσα Python. Διπλωματική Εργασία, τμήμα Εφαρμοσμένης Πληροφορικής, Πανεπιστήμιο Μακεδονίας

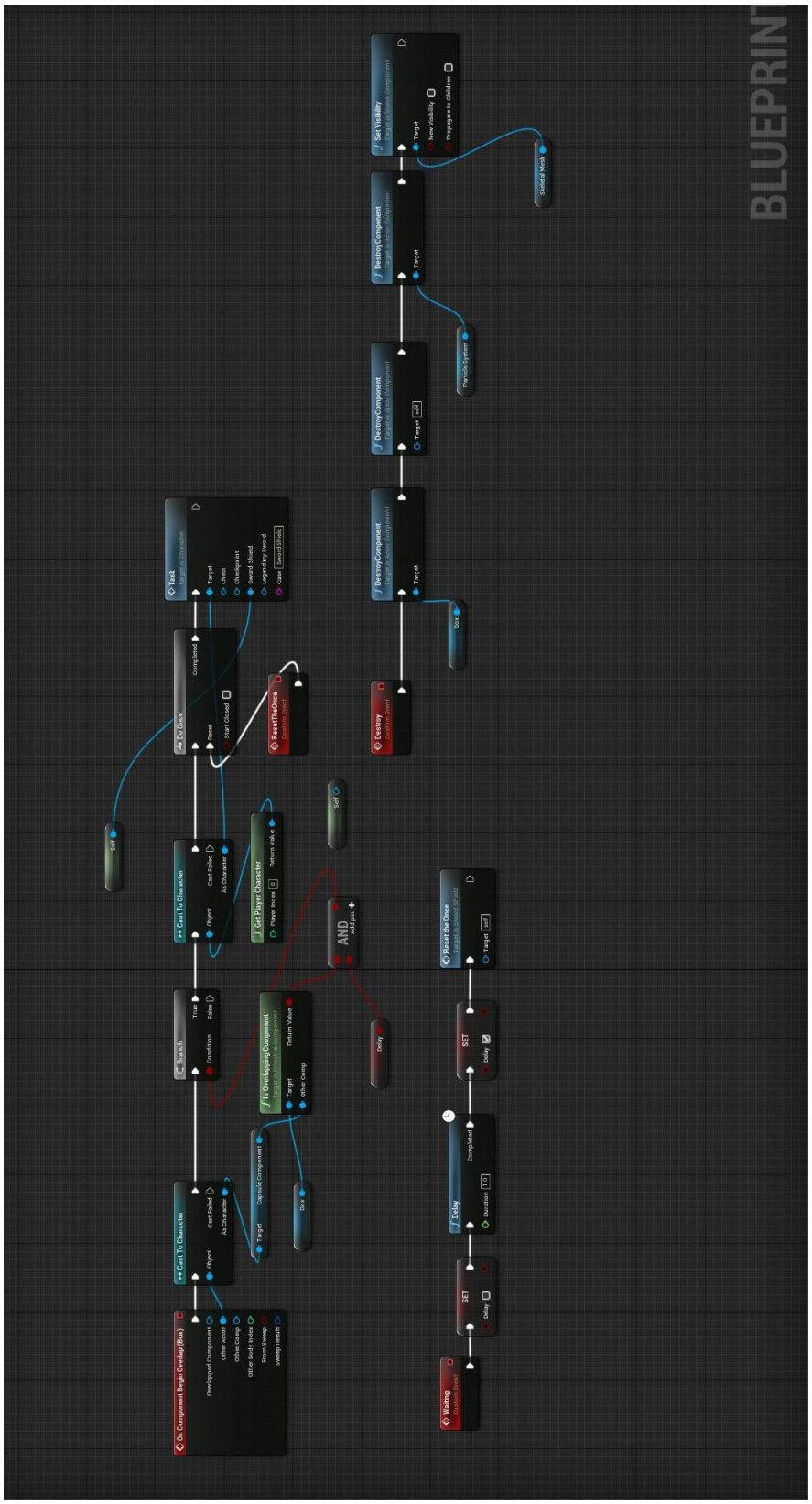
Παράρτημα



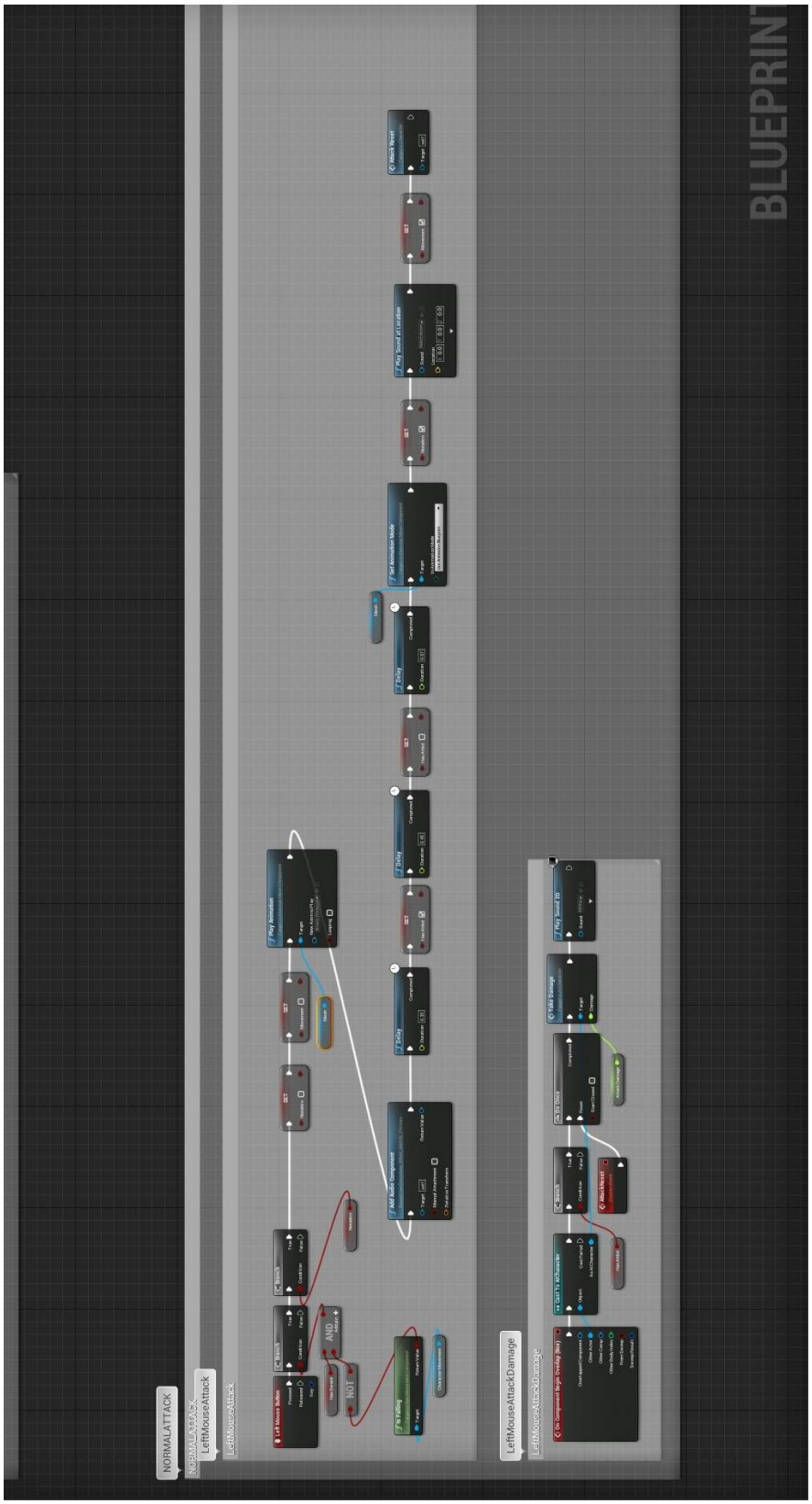
Blueprint 3: Γράφος ροής επιλογής "Run"



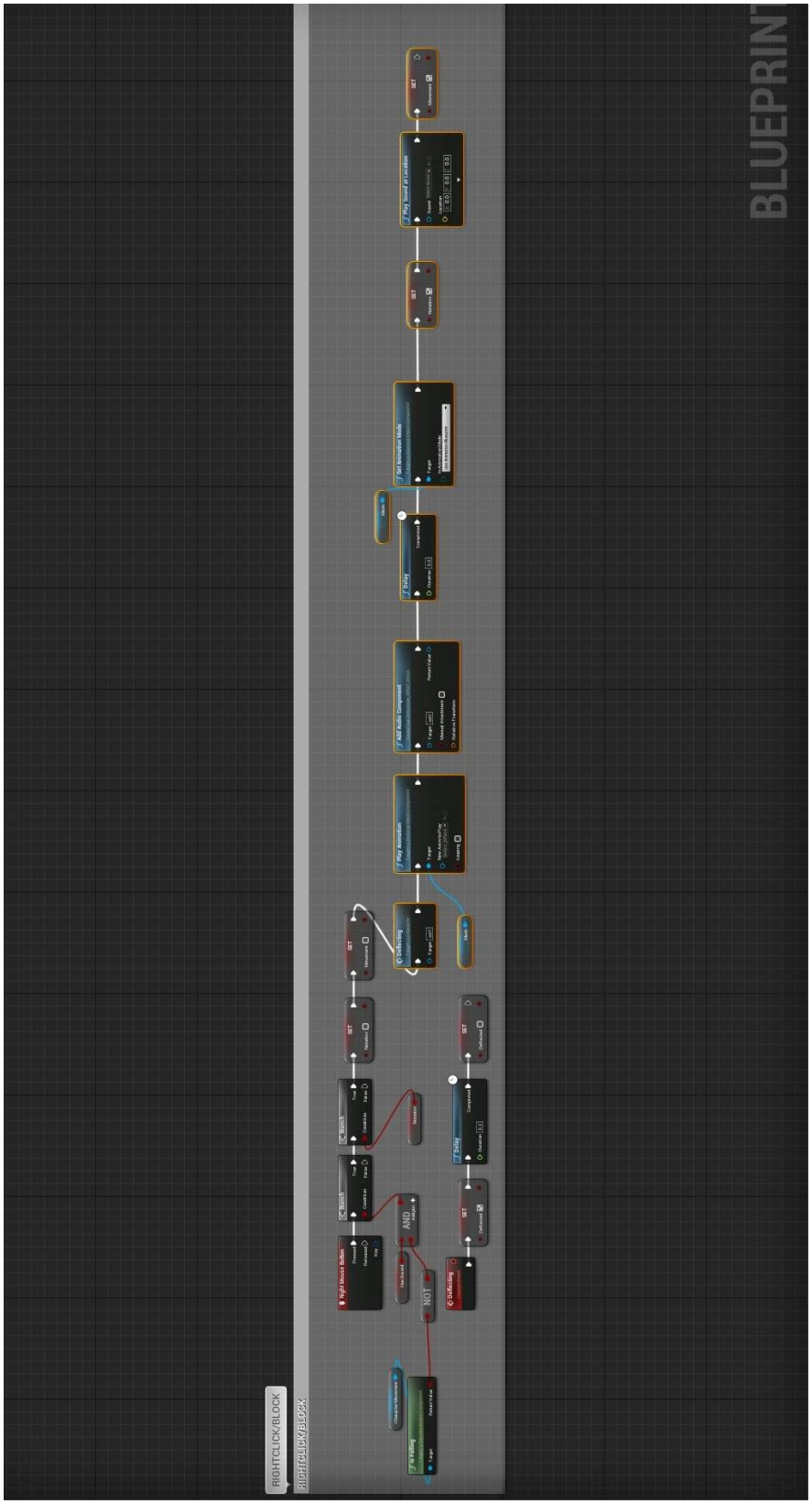
Blueprint 4: Γράφος ροής επιλογής θεωρίας



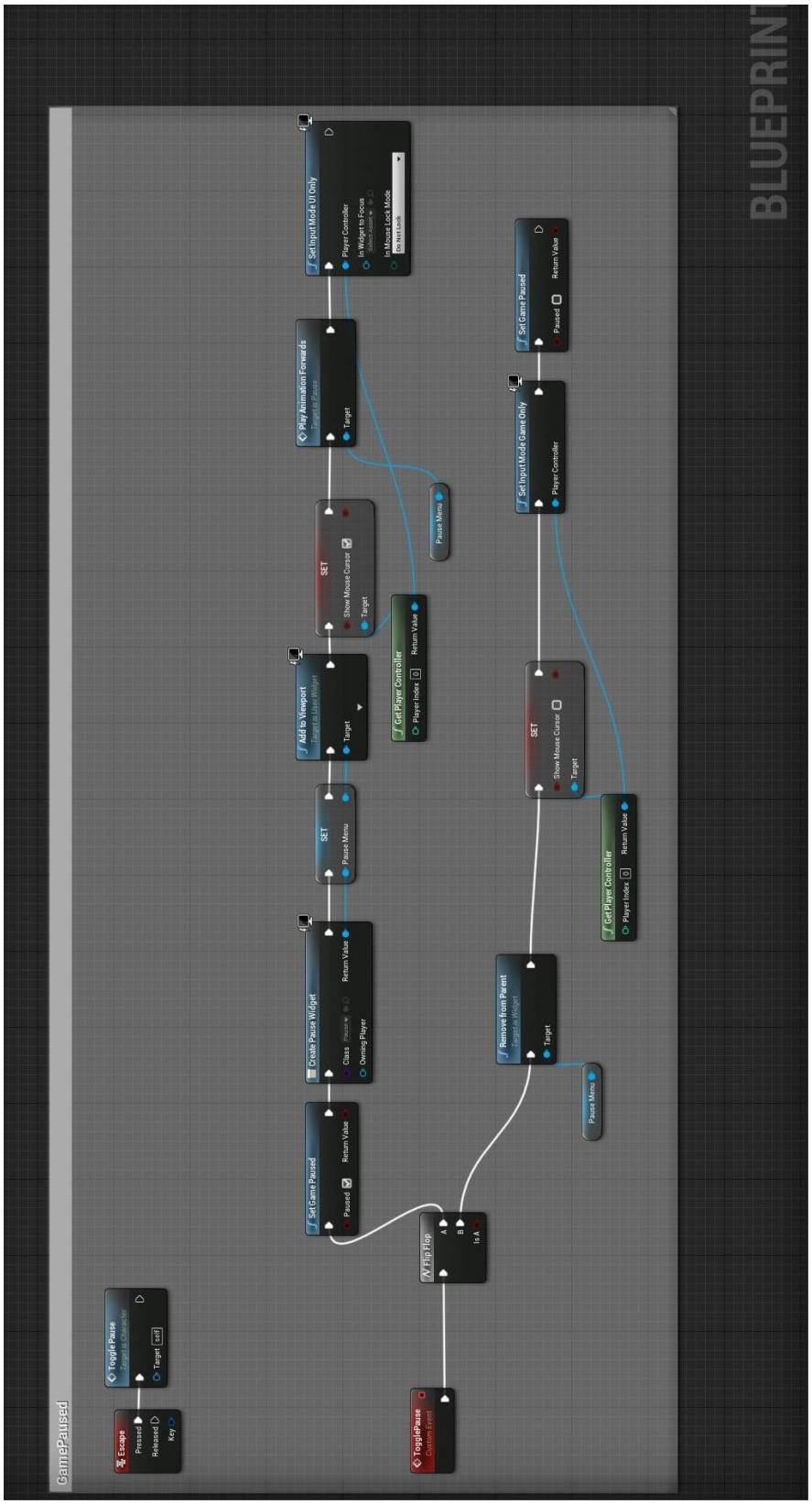
Blueprint 5: Γράφος ροής επιλογής σημείου ενδιαφέροντος



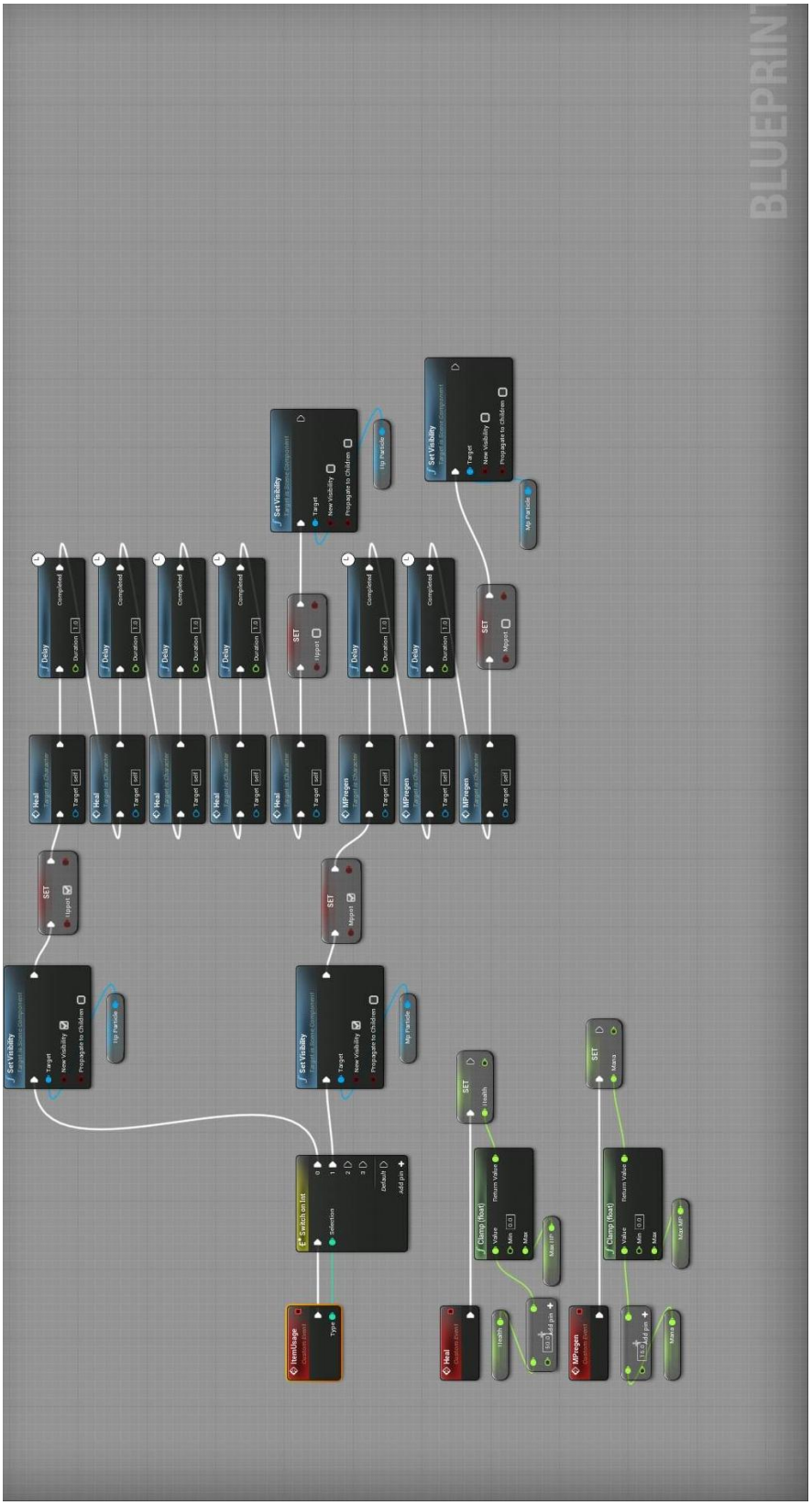
Blueprint 6: Βασική επίθεση χαρακτήρα



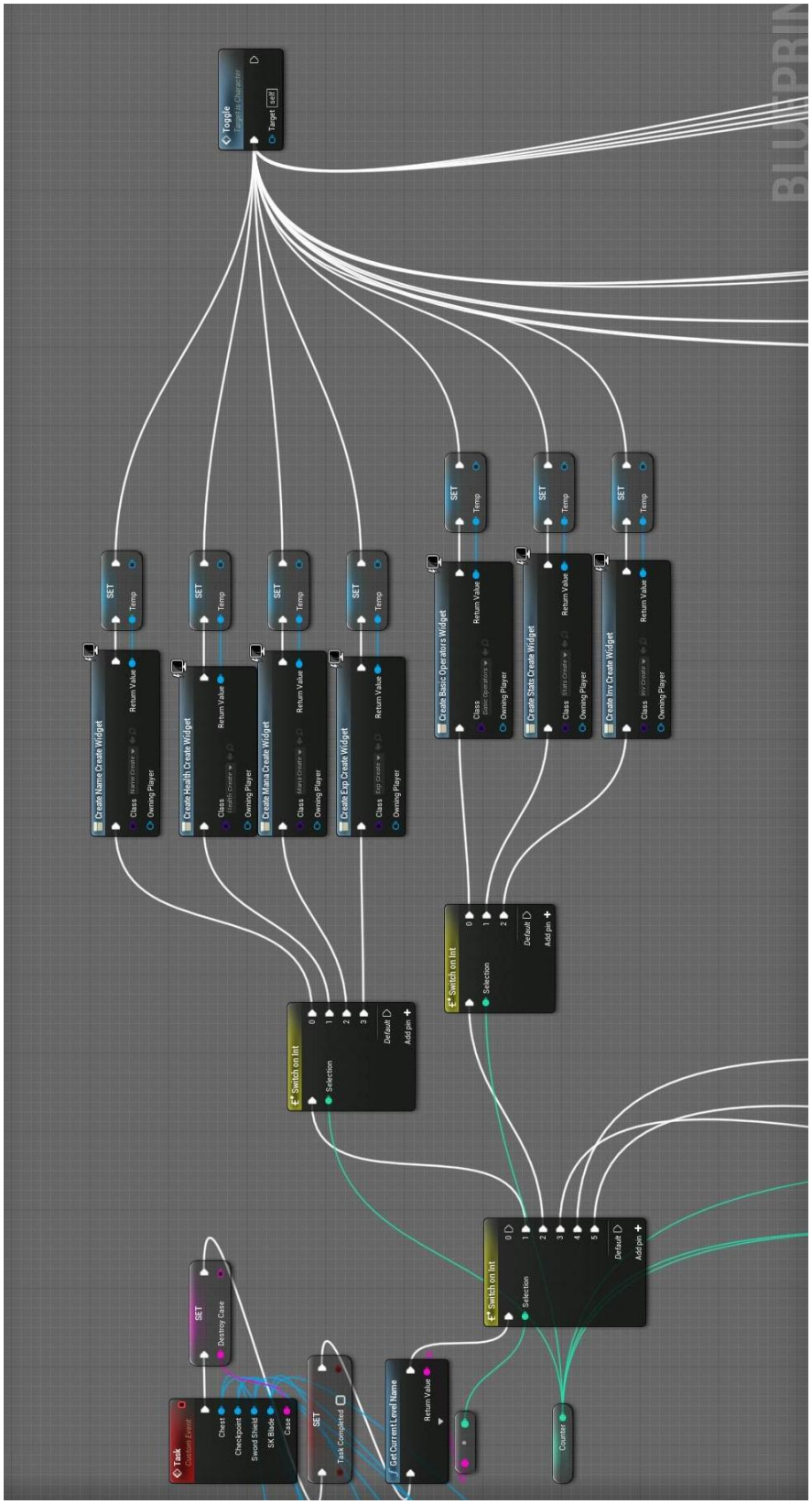
Blueprint 7: Απόκρουση επιθέσεων από τον χαρακτήρα



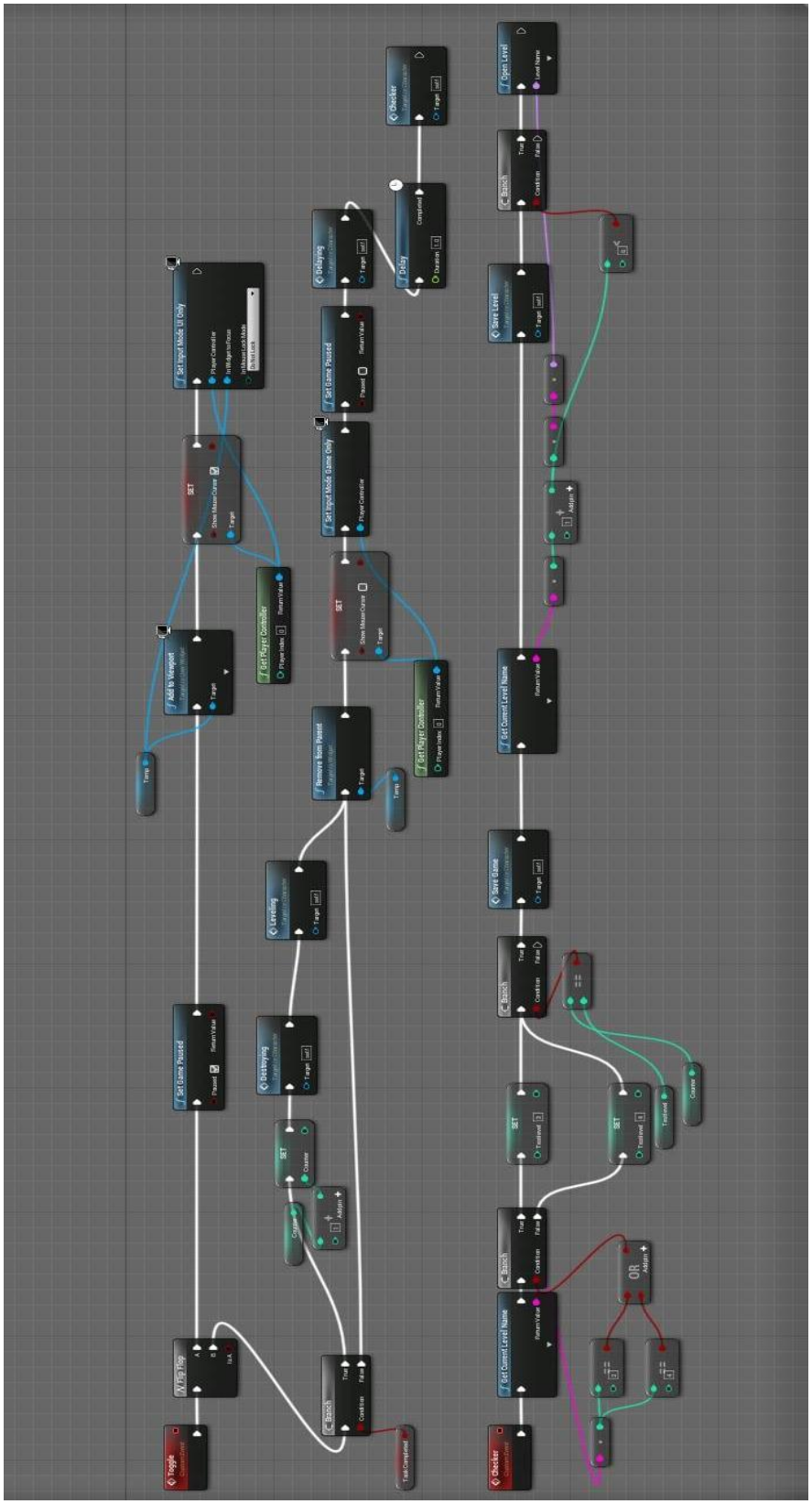
Blueprint 8: Δημιουργία μενού "Pause" και "πάγωμα" του παιχνιδιού



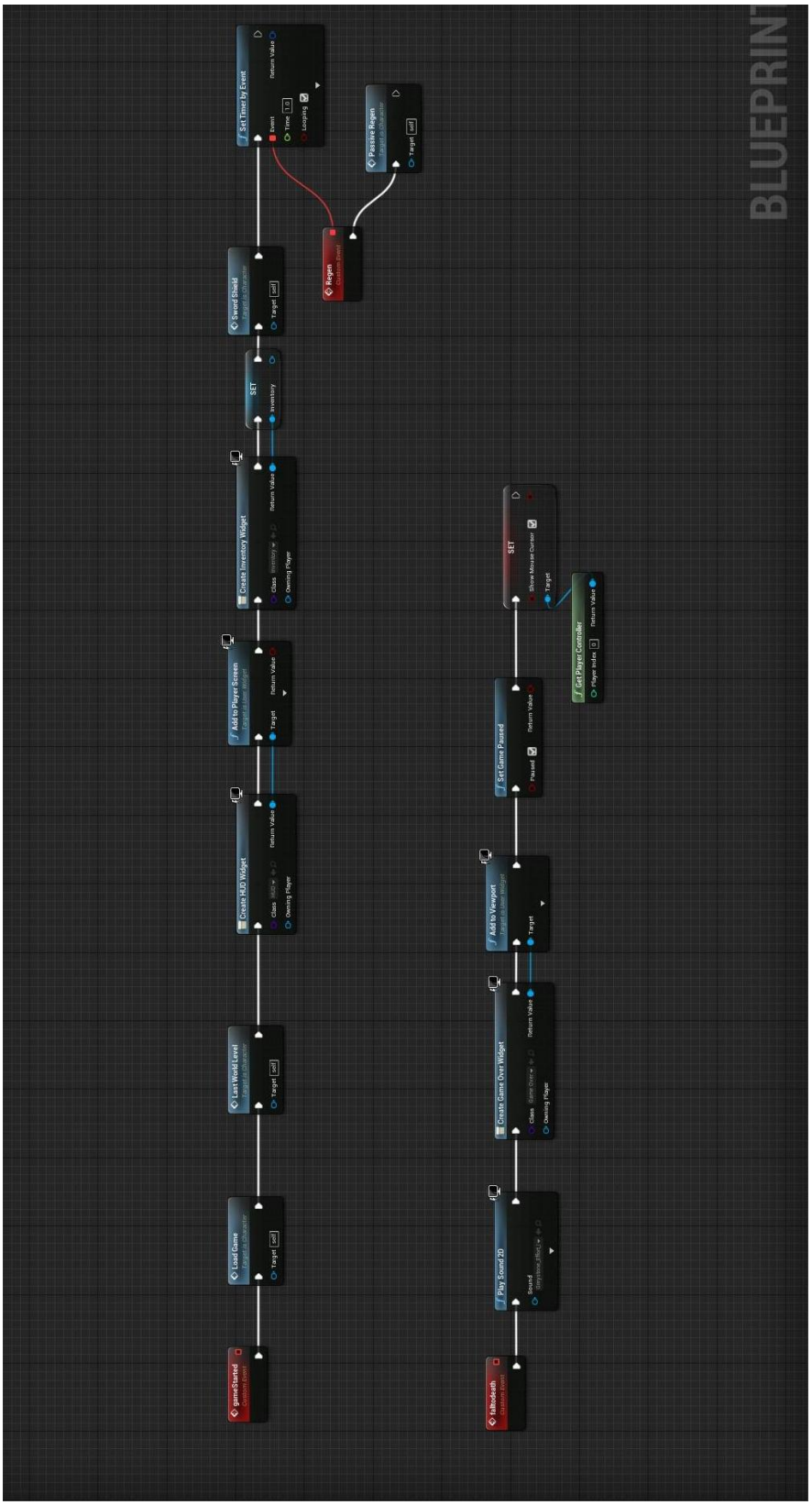
Blueprint 9: Χρήση φίλτρων από τον ήρωα



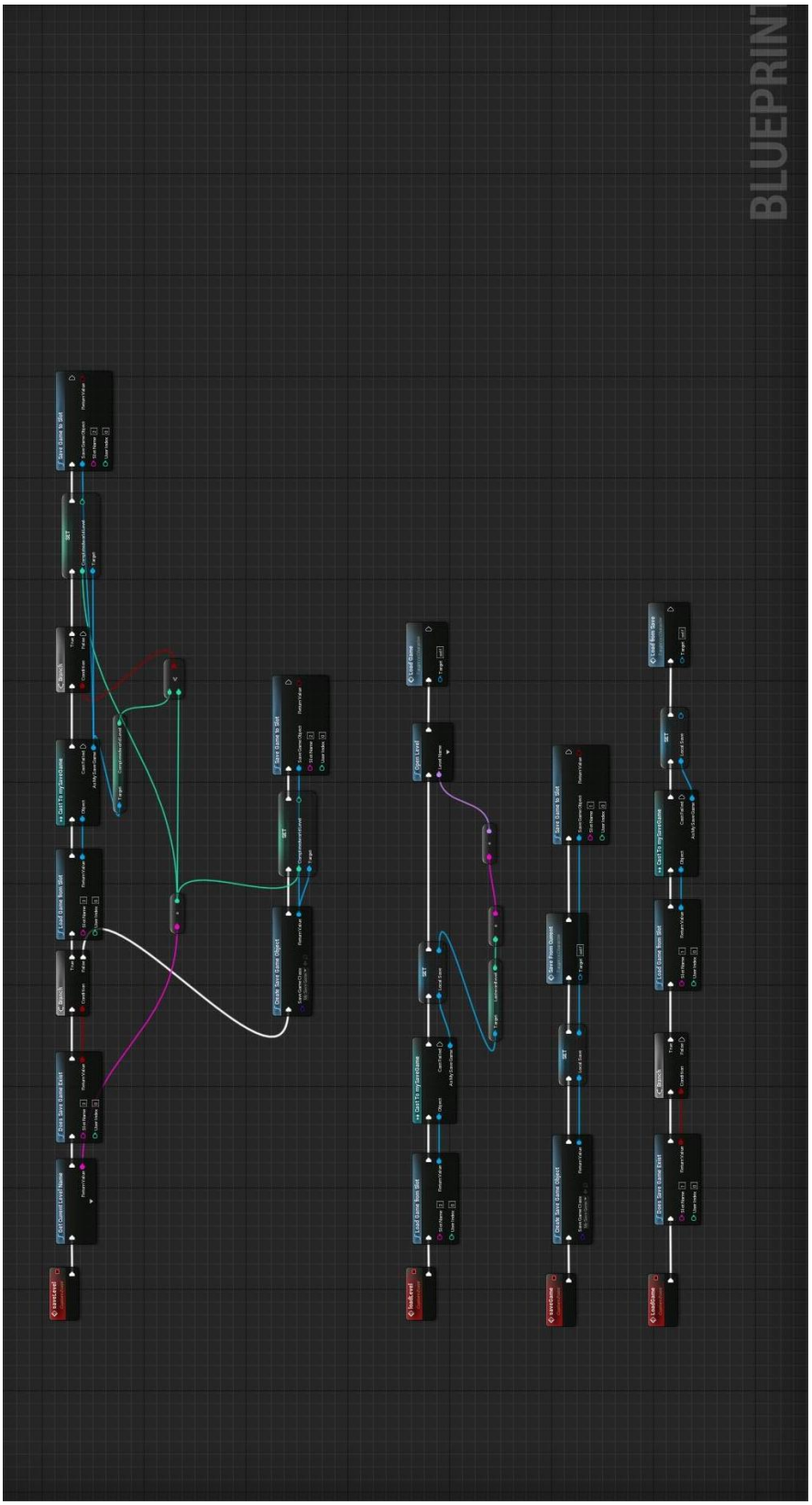
Blueprint 10: Εμφάνιση δοκιμασίας στην οθόνη



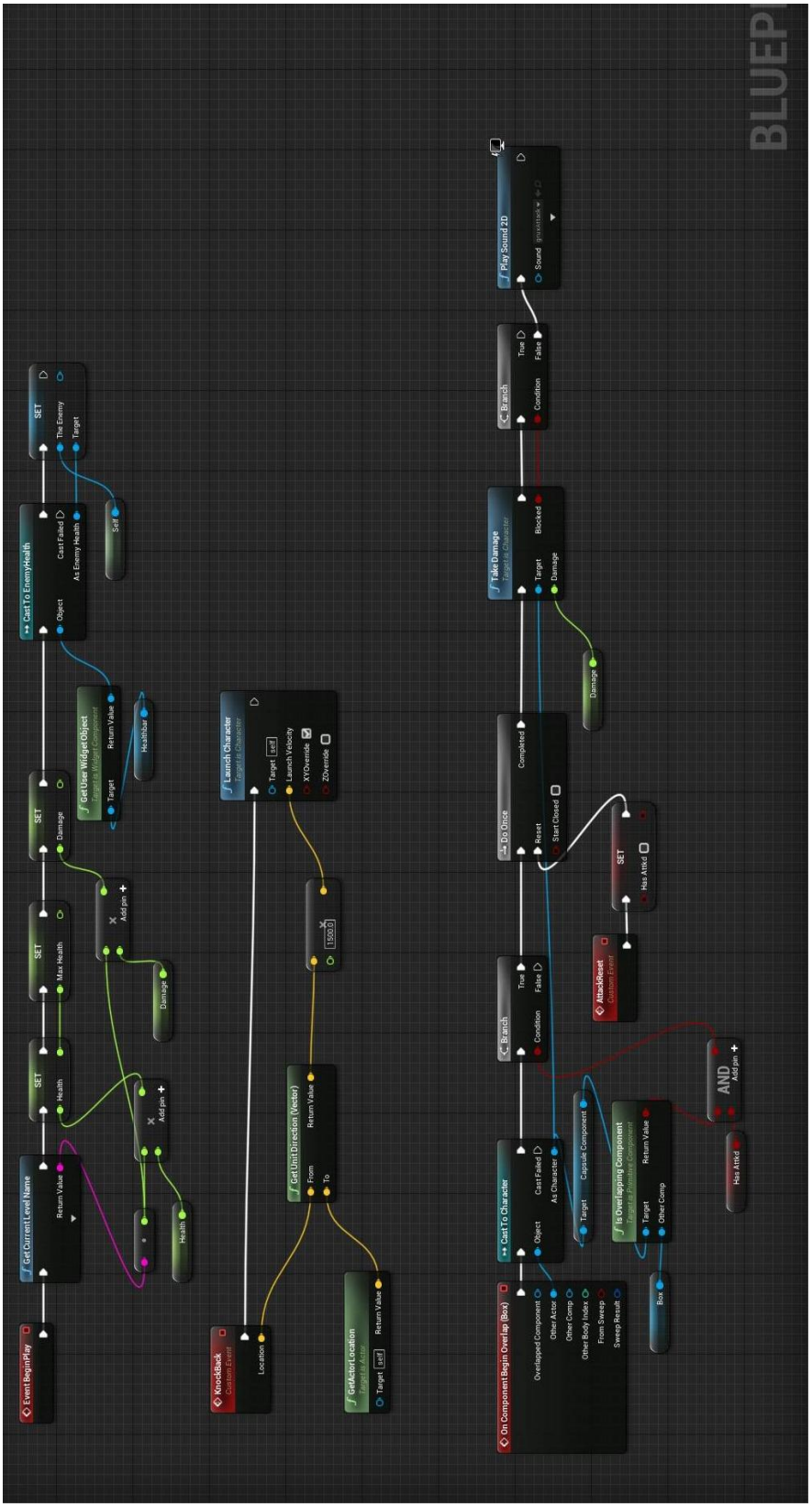
Blueprint 11: Event "Toggle" και "Checker"



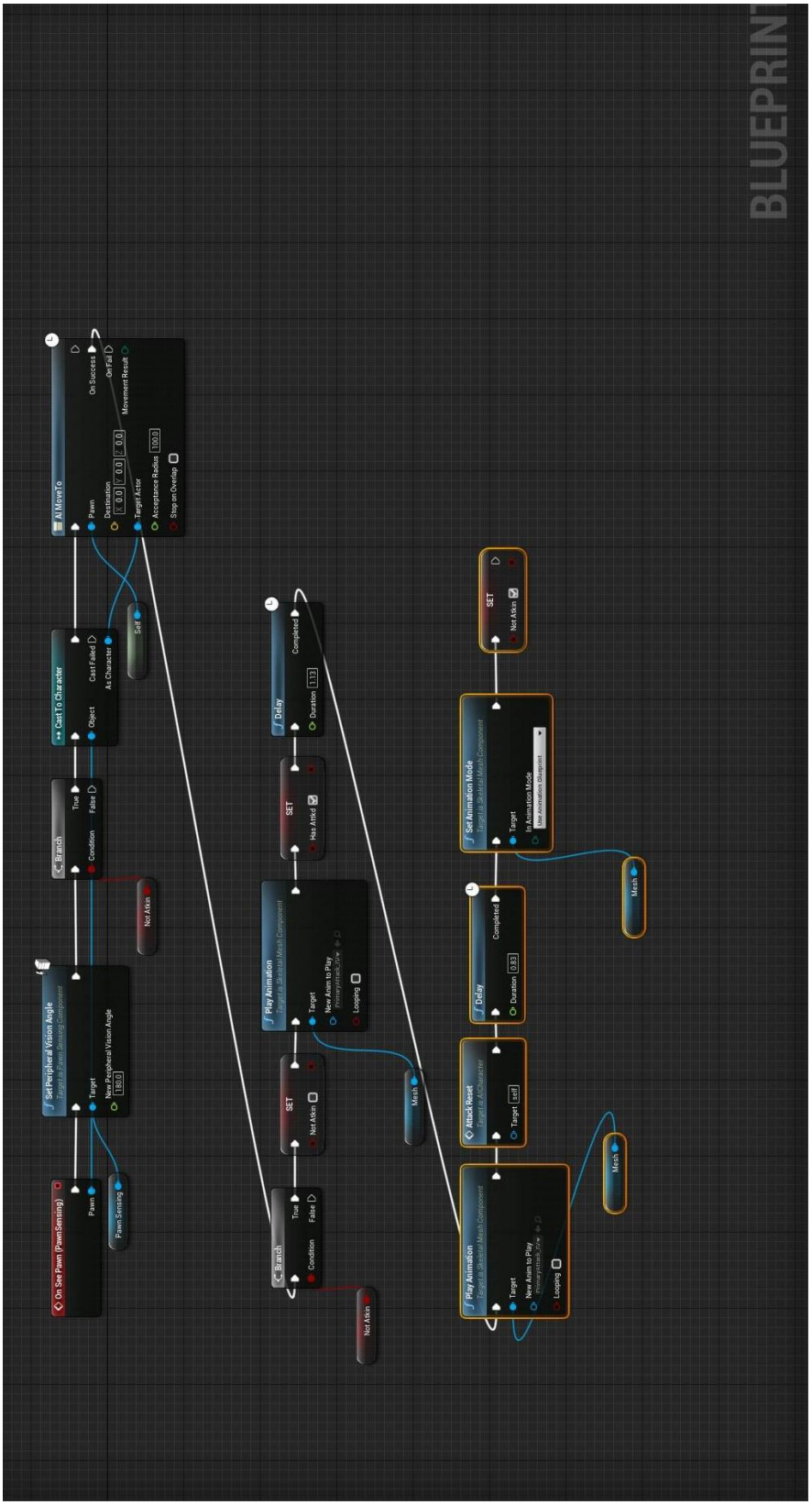
Blueprint 12: Event "Game Started" και "FalltoDeath"



Blueprint 13: Event "SaveLevel", "SaveGame", "LoadLevel", "Load Game"



Blueprint 14: Event και ροή γράφου εχθρού



Blueprint 15: Ροή γράφου "PawnSensing" εχθρού