|  |  |
| --- | --- |
| **logo-up-4color-stamp** | **ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ**  **ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΤΕΧΝΟΛΟΓΙΑΣ ΥΠΟΛΟΓΙΣΤΩΝ**  **ΤΟΜΕΑΣ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ & ΤΕΧΝΟΛΟΓΙΑΣ ΤΗΣ ΠΛΗΡΟΦΟΡΙΑΣ** |

**ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ**

**«ΑΝΑΠΤΥΞΗ ΑΛΛΗΛΕΠΙΔΡΑΣΤΙΚΟΥ ΜΥΘΙΣΤΟΡΗΜΑΤΟΣ ΜΕ ΣΥΓΧΡΟΝΑ ΕΡΓΑΛΕΙΑ»**

ΔΕΡΒΙΣΗΣ ΧΡΗΣΤΟΣ

1057744

**ΕΠΙΒΛΕΠΩΝ : ΚΥΡΙΑΚΟΣ ΣΓΑΡΜΠΑΣ**

ΠΑΤΡΑ,

Πανεπιστήμιο Πατρών, Τμήμα Ηλεκτρολόγων Μηχανικών και Τεχνολογίας Υπολογιστών.

Δερβίσης Χρήστος

© 2024 – Με την επιφύλαξη παντός δικαιώματος

Το σύνολο της εργασίας αποτελεί πρωτότυπο έργο, παραχθέν από τον Δερβίση Χρήστο και

δεν παραβιάζει δικαιώματα τρίτων καθ’ οιονδήποτε τρόπο. Αν η εργασία περιέχει υλικό, το οποίο δεν έχει παραχθεί από τον ίδιο, αυτό είναι ευδιάκριτο και αναφέρεται ρητώς εντός του κειμένου της εργασίας ως προϊόν εργασίας τρίτου, σημειώνοντας με παρομοίως σαφή τρόπο τα στοιχεία

ταυτοποίησής του, ενώ παράλληλα βεβαιώνει πως στην περίπτωση χρήσης αυτούσιων γραφικών

αναπαραστάσεων, εικόνων, γραφημάτων κ.λπ., έχει λάβει τη χωρίς περιορισμούς άδεια του κατόχου των πνευματικών δικαιωμάτων για την συμπερίληψη και επακόλουθη δημοσίευση του υλικού αυτού.

# πιστοποιηση

Πιστοποιείται ότι η Διπλωματική Εργασία με τίτλο

ΑΝΑΠΤΥΞΗ ΑΛΛΗΛΕΠΙΔΡΑΣΤΙΚΟΥ ΜΥΘΙΣΤΟΡΗΜΑΤΟΣ ΜΕ ΣΥΓΧΡΟΝΑ ΕΡΓΑΛΕΙΑ

του φοιτητή του τμήματος Ηλεκτρολόγων Μηχανικών και Τεχνολογίας Υπολογιστών

ΔΕΡΒΙΣΗ ΧΡΗΣΤΟΥ ΤΟΥ ΑΘΑΝΑΣΙΟΥ

Αριθμός Μητρώου: 1057744

Παρουσιάστηκε στο Τμήμα Ηλεκτρολόγων Μηχανικών

# ΠΕΡΙΛΗΨΗ

Ανάπτυξη Αλληλεπιδραστικού Μυθιστορήματος με Σύγχρονα Εργαλεία

Ονοματεπώνυμο Φοιτητή: Δερβίσης Χρήστος

Ονοματεπώνυμο Επιβλέποντος: Σγάρμπας Κυριάκος

Το αλληλεπιδραστικό μυθιστόρημα (interactive fiction) αποτελεί μια διαχρονική μορφή ψυχαγωγίας, ακόμη και πριν την εποχή της διάδοσης του ηλεκτρονικού υπολογιστή, υπό τη μορφή βιβλίων και επιτραπέζιων παιχνιδιών. Κύριο γνώρισμα του συγκεκριμένου μέσου είναι η εξουσιοδότηση που παρέχει ο δημιουργός στον παίκτη-αναγνώστη να επηρεάσει σε μικρό ή μεγάλο βαθμό τη ροή και την έκβαση της πλοκής, υπό την οπτική γωνία του πρωταγωνιστή. Από φαινομενικά ασήμαντες λεπτομέρειες, έως και μη αναστρέψιμα γεγονότα που θα καθορίσουν το τέλος της ιστορίας, ο χρήστης καλείται να πάρει αποφάσεις, να ακολουθήσει στρατηγικές, να συναναστραφεί με χαρακτήρες και να ενεργήσει εκ μέρους του πρωταγωνιστή, κατά την κρίση του. Ένα αξιοσημείωτο πλεονέκτημα της αλληλεπιδραστικής αφήγησης είναι ότι βασίζεται στη φαντασία του δημιουργού και του παίκτη, γεγονός που σημαίνει ότι δε χρειάζεται απαραίτητα να βασιστεί σε εξωτερικούς πόρους, είτε πολύπλοκα συστήματα. Το παραπάνω χαρακτηριστικό καθιέρωσε το αλληλεπιδραστικό μυθιστόρημα ως μια από τις πρώτες διαδεδομένες μορφές παιχνιδιών ηλεκτρονικού υπολογιστή, σε μια εποχή όπου η αρχιτεκτονική των υπολογιστών υφίστατο περιορισμούς και η ανάπτυξη των γραφικών και του ήχου δεν είχαν ακόμη σημειωθεί. Ωστόσο, με την πάροδο των χρόνων, η εξέλιξη των υπολογιστικών συστημάτων δημιούργησε την ευκαιρία για την ανάπτυξη παιχνιδιών που εστίαζαν σε άλλα σημεία, όπως στην πιο παραστατική οπτικοακουστική εμπειρία και στη δυνατότητα παιχνιδιού με πολλαπλούς παίκτες (multiplayer), με αποτέλεσμα τα παιχνίδια αλληλεπιδραστικών μυθιστορημάτων να απομακρυνθούν από το προσκήνιο, τουλάχιστον υπό την πρώιμη μορφή τους. Παρ’ όλα αυτά, η επιρροή τους στα ηλεκτρονικά παιχνίδια είναι αισθητή έως και σήμερα, ενώ παράλληλα εξακολουθεί να υπάρχει μια δυναμική διαδικτυακή κοινότητα, της οποίας τα μέλη κοινοποιούν και μοιράζονται τα έργα τους, ενώ επίσης μεριμνούν για την ανάπτυξη και συντήρηση σύγχρονου λογισμικού που επιτρέπει τη δημιουργία αλληλεπιδραστικών μυθιστορημάτων.

Στόχος της παρούσας εργασίας είναι να πραγματοποιηθεί μια εισαγωγή και εξοικείωση με τα σύγχρονα εργαλεία που χρησιμοποιούνται στο χώρο του interactive fiction (authoring systems, editors, γλώσσες προγραμματισμού) αλλά και να πραγματοποιηθεί μια ουσιαστική επαφή με τον αντικειμενοστραφή προγραμματισμό, με απώτερο σκοπό τη δημιουργία ενός ανάλογου έργου. Παράλληλη επιθυμία είναι να αναδειχθεί πώς η ανάπτυξη ενός απλού εν γένει τέτοιου παιχνιδιού μπορεί να αποτελέσει αξιόλογη μέθοδο εκμάθησης προγραμματιστικών εννοιών.

# extensive english summary

**Development of Interactive Fiction Using Modern Tools**

Student Name: Christos Dervisis

Supervisor Name: Kyriakos Sgarbas

Interactive fiction is a timeless form of entertainment, even before the advent of the computer era, taking the form of books and board games. A key feature of this medium is the power it grants the player-reader to influence the flow and outcome of the plot to a small or large degree, from the protagonist's perspective. From seemingly insignificant details to irreversible events that will determine the story's ending, the user is called to make decisions, follow strategies, interact with characters, and act on behalf of the protagonist according to their judgment. One notable advantage of interactive storytelling is that it relies on the imagination of both the creator and the player, meaning it doesn’t necessarily need to depend on external resources or complex systems. This characteristic established interactive fiction as one of the earliest popular forms of computer games, when computer architecture was limited and graphics and sound effects were not yet advanced. However, over the years, the evolution of computing systems created the opportunity for games that focused on other aspects, such as a more immersive audiovisual experience and multiplayer gameplay, which led to interactive fiction games gradually fading from the spotlight, at least in their early form. Nevertheless, their influence on computer games is still evident today. At the same time, there continues to be a vibrant online community whose members share their works and ensure the development and maintenance of modern software that allows the creation of interactive fiction.

This thesis aims to introduce and familiarize readers with the modern tools used in the field of interactive fiction (authoring systems, editors, programming languages) and to offer a meaningful introduction to object-oriented programming, with the ultimate goal of creating such a game. A side objective is to highlight how the development of a relatively simple game of this genre can serve as a valuable method of learning programming concepts.

# ΠΕΡΙΕΧΟΜΕΝΑ

Κατάλογος Εικόνων - Πινάκων 10

1. Εισαγωγή 12

1.1 Κλασική Προσέγγιση του Interactive Fiction …………………………………………………… 12

1.2 Εναλλακτική Προσέγγιση……………………………………………………………………………….. 14

1. H Ιστορία του IF 14
   1. Η εποχή του *Adventure* και της Infocom 14
   2. Μετάβαση στην Οικιακή Ανάπτυξη 15
   3. Νέα Εποχή – Νέα Προσέγγιση 16
2. Authoring Systems 18
3. Η Κοινότητα του Interactive Fiction 19
4. Το Εργαλείο της Inkle 20
   1. Πλεονεκτήματα της Ink 20
      1. Ευκολία εκμάθησης 21
      2. Περιβάλλον και Εργαλεία 21
         1. Inky Editor 21
         2. Υποστήριξη 22
   2. Σύνταξη 23
      1. Τα Βασικά Μέρη 23
      2. Weaves 27
      3. Μεταβλητές και Λογική 29
      4. Ροή Ελέγχου (Flow Control) 35
      5. Παρακολούθηση Κατάστασης (State Tracking) 37
      6. Διεθνής υποστήριξη χαρακτήρων 39
5. Το περιβάλλον της Unity 39
   1. Unity Editor 40
      1. Scene View (Όψη σκηνής) 40
      2. Hierarchy View (Ιεραρχία) 40
      3. Project Window (Παράθυρο Εργασίας) 41
      4. Inspector Window (Παράθυρο Επιθεώρησης) 41
   2. Η Φιλοσοφία του Αντικειμενοστραφούς Προγραμματισμού 42
      1. Αντικείμενα 43
      2. InkManager 44
      3. Κλάσεις 45
      4. Prefabs 45
      5. Scriptable Objects 46
      6. Πολυμορφισμός 49
6. Ενσωμάτωση στη Unity 49
   1. Προεργασία 50
   2. Χρόνος Εκτέλεσης και API 52
      1. Μετατροπή .ink σε .json 52
      2. Ροή της ιστορίας 53
      3. Αποθήκευση - Φόρτωση 55
   3. Ετικέτες (Tags) 57
   4. Μεταβάσεις μέσω API 59
   5. Αλληλεπίδραση με Μεταβλητές και Λίστες 61
      1. Η συνάρτηση DisplayAttributes() 61
      2. Η συνάρτηση SpendXP() 62
      3. Η συνάρτηση DisplayStatements() 63
7. Αθέμιτες Πρακτικές σε Unity Projects 64
   1. Προβλήματα Επίδοσης 64
      1. Λανθασμένη διαχείριση της μεθόδου Update() 64
      2. Η μέθοδος Find() 65

8.2 Προβλήματα Συντήρησης ………………………………………………………………………………. 66

1. Το εργαλείο Midjourney 69
2. Το παιχνίδι *Beyond Man* 72
   1. Η φιλοσοφία του παιχνιδιού 72
   2. Σύνοψη του παιχνιδιού 73
   3. Έλεγχος Περατότητας 74
3. Επίλογος 75

ΠAΡΑΡΤΗΜΑ

76

ΒΙΒΛΙΟΓΡΑΦΙΑ

77

# Κατάλογος Εικόνων – Πινάκων

ΕΙΚΟΝΕΣ

1. Κλασική μορφή παιχνιδιού IF (Πηγή: Jonathan Kotchian – Georgia Institute of Technology).. 12
2. Παράδειγμα ενός απλού parsing tree για τον υπολογισμό μιας μαθηματικής έκφρασης (Πηγή: TechTarget) . 13
3. Στιγμιότυπο του Adventure (Πηγή: Microsoft Store). 15
4. Συγγραφή κώδικα σε TADS 3.1 (Πηγή: StackOverflow) 16
5. Twine storyboard (Πηγή: The City University of New York).. 17
6. Κατάλογος έργων IF στο IF Database (Πηγή: The Interactive Fiction Database) 20
7. Απόσπασμα της λίστας διαγωνισμών IF (Πηγή: The Interactive Fiction Database).. 20
8. Το interface τoυ Inky editor.. 23
9. Η συνάρτηση fight( ).32
10. Το stitch fight\_figure. 33
11. Η συνάρτηση try( ).33
12. Κλήση της συνάρτησης try( ).34
13. Η συνάρτηση timepass( ).34
14. Στιγμιότυπο επιλογών που επηρεάζονται από την time\_of\_day. 35
15. Τα βασικά μέρη του Editor (Πηγή: Unity Technologies). 40
16. Ενδεικτικό απόσπασμα του Hierarchy View.. 41
17. Το παράθυρο Inspector για το αντικείμενο Save Button.. 42
18. Στιγμιότυπο του παιχνιδιού από ευρήματα του χαρακτήρα (Inventory Panel).. 43
19. To Hierarchy View τη στιγμή της εικόνας [18]. 44
20. Prefabs κειμένου και κουμπιών επιλογής.. 46
21. Η κλάση CharacterProfile που προέρχεται από την ScriptableObject.. 47
22. Δημιουργία Character Asset από το Project Window.. 48
23. Η κλάση CreateProfile.. 48
24. Οι καρτέλες των χαρακτήρων. 49
25. Ο φάκελος του πακέτου της Ink, εντός του φακέλου Assets.. 50
26. Η μέθοδος StartCoroutine.. 51
27. Κώδικας εκκίνησης της εφαρμογής.. 52
28. Κλήση της Continue() από την DisplayNextLines(). 54
29. Κλήση της Continue() από την WriteMoreLines(). 54
30. Η συνάρτηση DisplayChoices(). 55
31. Οι συναρτήσεις CreateSaveGameObject() και SaveGame().. 56
32. Η συνάρτηση LoadGame(). 56
33. Η συνάρτηση LoadState(). 57
34. Διαχειριστής σφαλμάτων.. 57
35. Χρήση tags στην Ink. 59
36. Η συνάρτηση Adjust().. 59
37. Η συνάρτηση CheckForInput.. 60
38. Η συνάρτηση ReadString(). 60
39. Απόσπασμα της συνάρτησης DisplayAttributes(). 61
40. Η συνάρτηση SpendXP(). 62
41. Στιγμιότυπο του παιχνιδιού με την καρτέλα ‘Attributes’.. 63
42. Η συνάρτηση DisplayStatements(). 64
43. Παράδειγμα κακής χρήσης της Update(). (Πηγή: Antonio Borrelli, Gerardo Canfora, Vittoria Nardone, Guiseppe A. Di Lucca, Massimiliano Di Penta - Detecting Video Game-Specific Bad Smells in Unity Projects ) 65
44. Αναφορά του αντικειμένου money στη σκηνή μέσω του Inspector.. 66
45. Χρήση του [SerializeField]. 68
46. Ανάθεση αντικειμένων στα πεδία που ορίζει το [SerializeField]. 68
47. Αποτελέσματα εικόνων του Midjourney (Πηγή: Curtis Pykes – DataCamp). 69
48. Εικονογραφία του παιχνιδιού.. 71
49. Εικονογραφία του παιχνιδιού.. 71
50. Εικονογραφία του παιχνιδιού.. 72

ΠΙΝΑΚΕΣ

1. Ταξινόμηση διαφόρων authoring tools 18

1. Εισαγωγή

1.1 Κλασική προσέγγιση του Interactive Fiction

Επίσημα, ένα έργο Interactive Fiction, αποτελεί ένα αλληλεπιδραστικό πρόγραμμα υπολογιστή. Παρ’ όλο που αυτό μπορεί να περιλαμβάνει γραφικά, ήχο και κινούμενες εικόνες, αναπόσπαστος κορμός του IF (Interactive Fiction) είναι η παρουσία κειμένου στην έξοδο (output) και, ως επί το πλείστον, εισαγωγή κειμένου στην είσοδο (input). Στην κλασική περίπτωση, ο παίκτης πληκτρολογεί μια φράση (command) σε μορφή εντολής (π.χ. “open the wooden door”). Το πρόγραμμα καλείται να αποσαφηνίσει την εντολή (εάν μπορεί) και να εξετάσει αν ο χαρακτήρας μπορεί να πραγματοποιήσει την αντίστοιχη ενέργεια. Η ερμηνεία των εντολών επιτυγχάνεται με δυο βασικά εργαλεία, τον αναλυτή (parser), και το world model.

|  |  |
| --- | --- |
| |  | | --- | | A screenshot of a text  Description automatically generated  Εικόνα 1: Κλασική μορφή παιχνιδιού IF (Πηγή: Jonathan Kotchian – Georgia Institute of Technology). | |

Ο parser είναι κατά κύριο λόγο τμήμα ενός υπολογιστικού μεταγλωττιστή (compiler), λαμβάνοντας ως είσοδο μια ακολουθία πηγαίου κώδικα, την οποία στη συνέχεια χωρίζει σε μικρότερα κομμάτια (tokens), τα οποία μπορεί να κατανοήσει ο compiler. Αυτό είναι το στάδιο της λεξικής ανάλυσης. Έπειτα ακολουθεί η συντακτική ανάλυση, όπου χρησιμοποιείται μια δενδροειδής δομή δεδομένων, το λεγόμενο parse tree/derivation tree. Ο συντακτικός αναλυτής χρησιμοποιεί τα tokens για να συνθέσει ένα parse tree, το οποίο συσχετίζει την προκαθορισμένη γραμματική της γλώσσας προγραμματισμού με τα tokens. Εάν εντοπίσει συντακτικό σφάλμα, ο αναλυτής το αναφέρει. Στη συνέχεια, πραγματοποιείται η σημασιολογική ανάλυση, όπου το parse tree επαληθεύεται βάσει ενός πίνακα συμβόλων, για να διαπιστωθεί ότι είναι σημασιολογικά έγκυρο. Η επαλήθευση γίνεται τόσο σε επίπεδο τύπου δεδομένων, όσο και σε ροή ελέγχου.

Προς αποφυγή παρεξηγήσεων, αξίζει να σημειωθεί ότι μερικές πηγές αναφέρουν ως parsing μόνο το στάδιο της συντακτικής ανάλυσης.

|  |
| --- |
| A screenshot of a computer  Description automatically generated  Εικόνα 2: Παράδειγμα ενός απλού parsing tree για τον υπολογισμό μιας μαθηματικής έκφρασης (Πηγή: TechTarget) . |
|  |
|  |

Το world model του προγράμματος είναι υπεύθυνο για την προσομοίωση των πραγμάτων που υπάρχουν στον εικονικό κόσμο της ιστορίας, τη συμπεριφορά τους και την διαδραστικότητά τους. Με λίγα λόγια, είναι το σύνολο κανόνων και εννοιών που διέπουν αυτό τον εικονικό κόσμο. Τα κύρια μέρη ενός world model είναι τα αντικείμενα (οbjects) και τα γεγονότα (events).

Τα objects ορίζουν τον υλικό κόσμο της ιστορίας, μπορούν να είναι οτιδήποτε, όπως δωμάτια, κλειδιά, χαρακτήρες και κάθε λογής αντικείμενο επιθυμεί να ορίσει ο δημιουργός. Κάθε object μπορεί να εμπεριέχει διαφόρων τύπων δεδομένα, συνήθως ιδιότητες (properties), όπως όνομα και περιγραφή και γνωρίσματα (attributes), όπως Boolean μεταβλητές που δηλώνουν συγκεκριμένα χαρακτηριστικά του αντικειμένου (π.χ. αν είναι όπλο, δοχείο κλπ). Αντικείμενα με attributes μπορούν να «κληρονομήσουν» (inherit) δεδομένα και μεθόδους (methods) από κλάσεις (classes). Αντίστοιχα, τα αντικείμενα μπορούν να έχουν και καταστάσεις (states) και συνθήκες (conditions), αμφότερα παρόμοια με τα attributes, αλλά πιο δυναμικοί τύποι δεδομένων. Για παράδειγμα, ένας διακόπτης μπορεί να είναι στη θέση on ή off, ένα κερί μπορεί να είναι αναμμένο ή σβηστό, μια ζυγαριά μπορεί να δείχνει 10, 65 ή 100. Τα objects οργανώνονται βάσει κληρονομικότητας (inheritance), δηλαδή μπορούν να είναι «γονείς» (parents), «παιδιά» (children) ή «αδέρφια» (siblings). Λόγου χάριν, δυο κλειδιά μέσα στο ίδιο κουτί έχουν για «γονέα» το κουτί και είναι μεταξύ τους «αδέρφια».

Τα events είναι απλούστερο να εξηγηθούν, καθώς είναι τα γεγονότα που συμβαίνουν κατά τη διάρκεια του παιχνιδιού και πώς αντιδρούν τα objects σε αυτά τα γεγονότα.

1.2 Εναλλακτική Προσέγγιση

Μια εναλλακτική σύγχρονη προσέγγιση του interactive fiction, είναι η περίπτωση όπου η ιστορία εξελίσσεται μέσω προκαθορισμένων επιλογών (choice based IF/Choose Your Own Adventure), από τις οποίες μπορεί να επιλέγει ο παίκτης. Στην προκειμένη περίπτωση, το παιχνίδι αποτελείται από ένα σύνολο κόμβων (nodes) και επιλογών, οι οποίες δρουν ως υπερσύνδεσμοι μεταξύ των κόμβων (hyperlinks). Συνήθως, κάθε γύρος περιλαμβάνει ένα κομμάτι κειμένου που περιγράφει την κατάσταση του παιχνιδιού εκείνη τη στιγμή. Σε ορισμένα κομμάτια, ακολουθεί μια σύντομη λίστα ευδιάκριτων επιλογών από τις οποίες ο παίκτης μπορεί να διαλέξει. Σε άλλα κομμάτια, ενδεχομένως να υπάρχουν υπερσύνδεσμοι ενσωματωμένοι σε διάφορα σημεία του κειμένου. Η ιστορία μεταβαίνει σε επόμενο στάδιο που παρουσιάζει περαιτέρω επιλογές ή υπερσυνδέσμους όταν ο παίκτης κάνει κλικ σε μια επιλογή ή έναν υπερσύνδεσμο. Το παιχνίδι συνεχίζεται με αυτόν τον τρόπο, μεταβαίνοντας ανάμεσα σε κόμβους της πλοκής, μέχρι να φτάσει σε ένα καταληκτικό κομμάτι της ιστορίας.

2 . H Ιστορία του IF

2.1 Η εποχή του *Adventure* και της Infocom

Η πορεία του Interactive Fiction ξεκίνησε τη δεκαετία του 1970, για τους περισσότερους συγκεκριμένα το 1977, με την κυκλοφορία του *Adventure*, αλλιώς γνωστό και ως *Colossal Cave Adventure*, δημιουργία του προγραμματιστών Will Crowther και Don Woods. Το  *Adventure* είναι ένα parser παιχνίδι εξερεύνησης σπηλαίων, στο οποίο ο παίκτης διαβάζει τις περιγραφές του περιβάλλοντος χώρου στο τερματικό του υπολογιστή και καλείται να πληκτρολογεί εντολές 2 λέξεων (ρήμα, ουσιαστικό), είτε και απλά ένα γράμμα για να προχωρήσει προς μια κατεύθυνση (πχ. “w” για να κατευθυνθεί δυτικά) με σκοπό να λύνει γρίφους και να πλοηγείται στη σπηλιά. Αξίζει να σημειωθεί ότι το *Adventure* αποτέλεσε το πρώτο αντικείμενο διπλωματικής εργασίας που αφορούσε σε παιχνίδι υπολογιστή (Mary Ann Buckles, 1985).

|  |
| --- |
| A screenshot of a computer  Description automatically generated  Εικόνα 3: Στιγμιότυπο του Adventure (Πηγή: Microsoft Store). |

Λίγο αργότερα, μια παρέα φοιτητών και εργαζόμενων του MIT δημιούργησαν ένα ανάλογο παιχνίδι εξερεύνησης με αναβαθμισμένο parsing και world modelling, με τον τίτλο *Zork*. Το συγκεκριμένο παιχνίδι γνώρισε ιδιαίτερα μεγάλη απήχηση, τόσο μεγάλη που κινητοποίησε τους δημιουργούς του να ιδρύσουν τη δική τους εταιρεία, με όνομα Infocom, η οποία για τα επόμενα χρόνια κυριάρχησε στην αγορά των IF παιχνιδιών στις Ηνωμένες Πολιτείες, με τίτλους όπως μια ανθολογία γύρω από τον πρωτότυπο τίτλο *Zork*, και νέους τίτλους, όπως *Deadline* (1982), *Enchanter* (1983), *A Mind Forever Voyaging* (1985), μεταξύ πολλών άλλων.

Ωστόσο, στα μέσα της δεκαετίας του 1980, η Infocom εξαγοράστηκε από την Activision, με αποτέλεσμα να παύσει η δραστηριοποίηση στο κομμάτι των text adventure παιχνιδιών, αφήνοντας ένα ξεκάθαρο κενό στη συγκεκριμένη αγορά.

2.2 Μετάβαση στην Οικιακή Ανάπτυξη

Οι τίτλοι κυκλοφοριών της Infocom άσκησαν μεγάλη επιρροή, τόσο σε ενθουσιώδεις παίκτες, όσο και σε προγραμματιστές, σε σημείο όπου ξεκίνησε μια προσπάθεια αντίστροφης μηχανικής (reverse engineering) για ανακατασκευή του κώδικα που ανέπτυξε η Infocom. Πράγματι, ο Graham Nelson, τότε φοιτητής του Πανεπιστημίου της Οξφόρδης, ανέπτυξε την Inform, μια γλώσσα προγραμματισμού, ειδικά για τη δημιουργία έργων IF. Η Inform, μαζί με τη χρήση της ευρέως διαθέσιμης εικονικής μηχανής (virtual machine) Z-machine (η οποία αναπτύχθηκε από τους ιδρυτές της Infocom) επέτρεψε στους οικιακούς χρήστες να γράψουν τα δικά τους interactive fiction έργα. Πρακτικά, o χρήστης για να παίξει τα παιχνίδια γραμμένα σε Inform μπορούσε απλώς να κατεβάσει έναν αναλυτή (interpreter) συμβατό με την πλατφόρμα του (λειτουργικό σύστημα και hardware). Άλλες εξίσου δημοφιλείς γλώσσες που δημιουργήθηκαν για τον ίδιο σκοπό ήταν η TADS (Text Adventure Development System) από τον Michael J. Roberts και η Hugo από τον Kent Tessman. Και οι τρεις παραπάνω γλώσσες έχουν έντονες ομοιότητες ως προς τη δομή και τον τρόπο συγγραφής και ακολουθούν την κλασική προσέγγιση του parsing, ενώ νέες εκδόσεις τους εξακολουθούν να αναπτύσσονται ακόμα και σε επίπεδο ανοιχτού κώδικα, προσδίδοντας παραπάνω δυνατότητες σε developers και παίκτες (π.χ. web play, dynamic compiling, multimedia support).

|  |
| --- |
| A computer screen shot of text  Description automatically generated  Εικόνα 4: Συγγραφή κώδικα σε TADS 3.1 (Πηγή: StackOverflow) |

2.3 Νέα Εποχή – Νέα Προσέγγιση

Γύρω στα τέλη της δεκαετίας του 2000 άρχισε να αναδύεται ένα νέο ρεύμα, το οποίο ξεκίνησε να αναθεωρεί την έννοια και την ως τότε δομή του interactive fiction. Το κλασικό parsing text adventure ήταν πλέον παρωχημένο για ένα μέρος της κοινότητας, τόσο ως προς την δυνατότητά του να παρουσιάζει παραστατικές και πλούσιες ιστορίες, όσο και ως προς το ενίοτε αφιλόξενο περιβάλλον που παρείχε σε developers και παίκτες. Υπό αυτό το πρίσμα, δημιουργήθηκαν νέα εργαλεία που εστίαζαν σε πιο εύχρηστα περιβάλλοντα και υιοθέτησαν κατά κύριο λόγο την choice-based προσέγγιση, δηλαδή σύνολο προσχεδιασμένων επιλογών και κόμβοι που συνδέονται μεταξύ τους με υπερσυνδέσμους.

Ένα επαναστατικό τέτοιο εργαλείο αποτέλεσε το Twine, το οποίο κυκλοφόρησε το 2009 από τον Chris Klimas. To Twine είναι μια web-based εφαρμογή, γραμμένη σε HTML5 και Javascript (υποστηριζόμενη και από CSS). Χαρακτηριστικό της είναι η έμφαση στην οπτική παρουσίαση του κειμένου, καθώς και το γεγονός ότι η συγγραφή ενός παιχνιδιού δε στηρίζεται σε κάποια προκαθορισμένη scripting γλώσσα προγραμματισμού, αλλά στα λεγόμενα “story formats”. Τα τελευταία αποτελούν frameworks, υπεύθυνα για τον τρόπο παρουσίασης της ιστορίας (διαμόρφωση κειμένου, εικόνων και λοιπών μέσων), τους κανόνες σχετικά με την εφαρμογή των επιλογών του παίκτη, την καταγραφή των μεταβλητών και την υποθετική λογική (conditional logic). Επίσης, επιτρέπουν την επεξεργασία της παρουσίασης της ιστορίας (μέσω HTML, CSS και JavaScript).

|  |
| --- |
| A screenshot of a computer  Description automatically generated  Εικόνα 5: Twine storyboard (Πηγή: The City University of New York). |

Υπάρχουν διάφορα story formats (Harlowe, SugarCube, Snowman, Chapbook), καθένα εκ των οποίων παρέχει διαφορετικές δυνατότητες και απευθύνεται σε δημιουργούς αντίστοιχου προγραμματιστικού επιπέδου . Μεγάλο πλεονέκτημα του Twine μπορεί να θεωρηθεί το γεγονός ότι απαιτεί ελάχιστη έως και μηδαμινή εμπειρία προγραμματισμού από το χρήστη που επιθυμεί να δημιουργήσει έστω μια απλή ιστορία.

Πολλά είναι τα παραδείγματα παρόμοιων εργαλείων, εκ των οποίων μερικά αξίζει να αναφερθούν. Το 2006, ο Graham Nelson παρουσίασε το Inform 7, μια επαναστατική μετεξέλιξη του αρχικού εργαλείου, με την ειδοποιό διαφορά ότι πλέον ο δημιουργός προγραμμάτιζε σε φυσική γλώσσα *("It's hard to imagine a more clear expression than ‘if every door in the Dining Room is closed’ as a way to ask if every door in the Dining Room is closed."*, Graham Nelson). Ένα ακόμη αξιοσημείωτο δημιούργημα είναι αυτό των Joe Humphries και Jon Ingold, ιδρυτών των Inkle Studios, οι οποίοι το 2011 παρουσίασαν την Ink, μια scripting γλώσσα που εξειδικεύεται στη συγγραφή choice-based ιστοριών και χρησιμοποιείται τόσο για κυκλοφορίες των Inkle Studios (*80 Days*, *Sorcery, A Highland Song*) όσο και από ανεξάρτητους δημιουργούς (*Bury Me My Love*, *Sable*, *Goodbye Volcano High*, *Neocab*). Εκτενής ανάλυση της Ink και του περιβάλλοντός της θα γίνει σε επόμενο κεφάλαιο, καθώς αφορά στην ανάπτυξη του παιχνιδιού της παρούσας εργασίας.

3. Authoring Systems

Έχει σημασία να αποσαφηνιστεί ο όρος «εργαλείο» για τη δημιουργία έργων IF. Τα μέσα που χρησιμοποιούνται στο σύνολό τους αποκαλούνται κατά την αγγλική βιβλιογραφία authoring systems, είτε authoring tools. Τυπικά, ένα authoring system αποτελείται από έναν αναλυτή (parser), προκαθορισμένη βιβλιοθήκη (library module) και μεταγλωττιστή (compiler). Το τελικό αποτέλεσμα του παραπάνω συστήματος είναι ένα αρχείο ιστορίας (story file), το οποίο στη συνέχεια θα πρέπει να φορτωθεί στο διερμηνέα (interpreter), προκειμένου να εκτελεσθεί από τον παίκτη.

Μερικά από τα πιο διαδεδομένα authoring systems παρατίθενται εδώ (ορισμένα αναφέρθηκαν παραπάνω):

* Choice-based: [ChoiceScript](https://www.ifwiki.org/ChoiceScript" \o "ChoiceScript), [Ink](https://www.ifwiki.org/Ink" \o "Ink), [Twine](https://www.ifwiki.org/Twine" \o "Twine).
* Parser-based: [ADRIFT](https://www.ifwiki.org/ADRIFT" \o "ADRIFT), [Adventuron](https://www.ifwiki.org/Adventuron" \o "Adventuron), [Dialog](https://www.ifwiki.org/Dialog" \o "Dialog), [Inform 6](https://www.ifwiki.org/Inform_6" \o "Inform 6), [Inform 7](https://www.ifwiki.org/Inform_7" \o "Inform 7), [TADS 3](https://www.ifwiki.org/TADS_3" \o "TADS 3), [ZIL](https://www.ifwiki.org/ZIL" \o "ZIL).
* Υβριδικού τύπου: [Quest](https://www.ifwiki.org/Quest_(Language)" \o "Quest (Language)).

Για σαφέστερη κατανόηση, παρακάτω ακολουθεί ένας πίνακας ταξινόμησης διάφορων authoring systems:

|  |  |  |  |
| --- | --- | --- | --- |
| Είσοδος | Μέσο | Authoring Tool | Σημειώσεις |
| Επιλογές  (Choice - based) | κείμενο | Οπτικός προγραμματισμός | Twine: Community project ανοικτού κώδικα, δημοσίευση στο διαδίκτυο |
| κείμενο | Scripting γλώσσα | Ink : Project ανοικτού κώδικα, κυκλοφορία των Inkle Studios |
| πολυμέσα | προγραμματισμός | Ren’ Py: Community project ανοικτού κώδικα, visual novels |
| Φυσική γλώσσα | κείμενο | markup γλώσσα | Inform: Community project ανοικτού κώδικα, μεταγλωττίζει σε z-code |
| κείμενο | Μ.Δ. | Infocom: Εμπορικά παιχνίδια, δημιουργία z-code αναλυτή |
| Πίνακας 1: Ταξινόμηση διαφόρων authoring tools. Μ.Δ.: Μη Διαθέσιμο | | | |

4. Η Κοινότητα του Interactive Fiction

Κατά τη δεκαετία του 1990, το interactive fiction αντιμετώπισε έντονη εμπορική κρίση, διότι τίτλοι νέας και διαφορετικής φιλοσοφίας κατείχαν το μεγαλύτερο μερίδιο της αγοράς και τα παιχνίδια που βασίζονταν σε απλό κείμενο φάνταζαν παρωχημένα. Ωστόσο, μια δυναμική και υπολογίσιμη κοινότητα φίλων και θαυμαστών των παιχνιδιών αλληλεπιδραστικών μυθιστορημάτων συσπειρώθηκε μέσα από διαδικτυακά forums και newsletters και το 1995 λήφθηκε η πρωτοβουλία για ένα διαγωνισμό μικρών IF ιστοριών. Ο διαγωνισμός ονομάσθηκε IFComp και η αναπάντεχα μεγάλη συμμετοχή ήταν αρκετή για να τον καθιερώσει ως ετήσιο θεσμό έως και σήμερα. Μάλιστα, από το 2014, γίνονται δεκτές και οι υποψηφιότητες ιστοριών που χρησιμοποιούν υπερκείμενο (hypertext), που ακολουθούν δηλαδή την choice - based προσέγγιση. Το παραπάνω αποδεικνύει την επιθυμία της κοινότητας να διευρύνει τους ορίζοντές της, να συμβαδίσει με την εποχή και αφήνει τα μέλη να προσδιορίζουν την ταυτότητά της. Ο διαγωνισμός IFComp αναφέρθηκε εκτενώς λόγω της ιστορικής του σημαντικότητας, όμως αξίζει να αναφερθεί ότι υπάρχει πληθώρα νεότερων διαγωνισμών που διοργανώνονται τακτικά και διατηρούν τα μέλη της κοινότητας δραστήρια. Οι διαγωνισμοί, καθώς και τα παιχνίδια αναρτώνται και μπορούν να βρεθούν στον ιστότοπο [www.ifdb.org](http://www.ifdb.org).

Εξίσου σημαντικό ρόλο στην κοινότητα κατέχει και το Interactive Fiction Technology Foundation, ένας μη κερδοσκοπικός οργανισμός που σχηματίστηκε το 2016 από σημαντικά μέλη της κοινότητας (μεταξύ αυτών ο Andrew Plotkin, από τους δημοφιλέστερους δημιουργούς έργων IF και ο Chris Klimas, δημιουργός του Twine) με σκοπό να υποστηρίξει κάθε μορφής ανάπτυξη δραστηριότητα γύρω από το ευρύτερο φάσμα του interactive fiction. Μάλιστα, από την ίδια χρονιά, το Interactive Fiction Technology Foundation έχει αναλάβει και τη διαχείριση του IFComp.

|  |
| --- |
| A screenshot of a computer  Description automatically generated  Εικόνα 6: Κατάλογος έργων IF στο IF Database (Πηγή: The Interactive Fiction Database) |
| A screenshot of a computer  Description automatically generated  Εικόνα 7: Απόσπασμα της λίστας διαγωνισμών IF (Πηγή: The Interactive Fiction Database). |

5. Το Εργαλείο της Inkle

5.1 Πλεονεκτήματα της Ink

Οι δημιουργοί της Ink ομολογουμένως θέλησαν να δημιουργήσουν ένα εργαλείο που να καλύπτει τις δημιουργικές και τεχνικές απαιτήσεις τους για την κατασκευή ιστοριών, καθιστώντας εύκολη τη μετάβαση μεταξύ των τμημάτων κειμένου και παρέχοντας τη δυνατότητα για ποικιλομορφία στο παιχνίδι. Πράγματι, η Inkle προσφέρει μια συλλογή από εργαλεία που ανταποκρίνονται στις ανάγκες των δημιουργών ιστοριών, ανεξάρτητα από το προγραμματιστικό υπόβαθρό τους .

5.1.1 Ευκολία εκμάθησης

Η Ink είναι μια scripting γλώσσα, η οποία συγκριτικά με γλώσσες εγγύτερα στον low-level προγραμματισμό (δηλαδή, εγγύτερα στο επίπεδο μηχανής), παρέχει τις εξής διευκολύνσεις:

* Έχει αρκετά απλή σύνταξη, παρόμοια σε μεγάλο βαθμό με τη φυσική γλώσσα. Πράγματι, η Ink επαληθεύει αυτό το χαρακτηριστικό, περισσότερο από πολλές άλλες γλώσσες (Για παράδειγμα, ένας τρόπος για να εξετάσει ο χρήστης εάν μια λίστα “inventory” περιέχει τη μεταβλητή “wallet” είναι να δηλώσει: inventory has wallet).
* Δεν υπάρχει ανάγκη για σαφή δήλωση του τύπου μιας μεταβλητής (π.χ. int, string, double κλπ), επομένως η συγγραφή είναι πιο αφαιρετική (Συνήθως, η δήλωση μεταβλητής στην Ink γίνεται γράφοντας μπροστά VAR, ανεξάρτητα αν εκφράζει boolean μεταβλητή, αριθμό ή κείμενο).
* Τέτοιες γλώσσες ασχολούνται ελάχιστα έως και καθόλου με πιο σύνθετες έννοιες που αφορούν στο hardware, όπως διαχείριση μνήμης και αρχείων και τύπους δεδομένων, αφήνοντας μεγαλύτερο περιθώριο στο χρήστη να ασχοληθεί με τη λογική δομή του προγράμματος.
* Το σύστημα ροής της ιστορίας και διακλαδώσεων της Ink είναι ιδιαίτερα εύληπτο, δηλαδή χρησιμοποιώντας και μόνο 4 απλούς συμβολισμούς ( ===, =, \*, ->) ο χρήστης είναι ικανός να συνθέσει ένα αρκετά απλό παιχνίδι.

5.1.2 Περιβάλλον και Εργαλεία

5.1.2.1 Inky Editor

Παρά το γεγονός ότι υπάρχουν επεκτάσεις της Ink και για συμβατικούς code editors (π.χ. VS Code), η Inkle έχει δημιουργήσει τον δικό της επίσημο editor, Inky. O συγκεκριμένος editor παρέχει στο συγγραφέα υψηλό επίπεδο αμεσότητας και διαφάνειας με τρόπους που παρουσιάζονται στη συνέχεια:

* Η διεπαφή της εφαρμογής χωρίζεται σε 2 παράθυρα. Στο αριστερό παράθυρο, ο χρήστης συντάσσει την ιστορία και στο δεξί βλέπει μια ταυτόχρονη προεπισκόπιση της ιστορίας του στην παρούσα κατάσταση, δηλαδή σε πραγματικό χρόνο (ream time). Αυτό συμβαίνει, επειδή η Ink είναι μια διερμηνευμένη γλώσσα (interpreted language), οπότε ο χρήστης «τρέχει» την ιστορία όσο τη γράφει, χωρίς να απαιτείται compiling. Επίσης, η διάταξη στο δεξί παράθυρο προϊδεάζει το χρήστη για το πώς περίπου θα φαίνεται το παιχνίδι εάν «τρέξει» σε browser.
* Η Ink παρέχει αυτόματα επισημάνσεις στη σύνταξη (syntax highlighting) σε πραγματικό χρόνο, δηλαδή διαφοροποιεί απευθείας τα διάφορα τμήματα του κώδικα (επιλογές, μεταβλητές, συνθήκες), κάνοντας τον κώδικα αρκετά πιο σαφή και ευδιάκριτο. Επιπλέον, επισημαίνει αμέσως συντακτικά σφάλματα.

5.1.2.2 Υποστήριξη

Το επίσημο εγχειρίδιο της γλώσσας (documentation) είναι ιδιαίτερα ευανάγνωστο και επεξηγεί αναλυτικά και παραστατικά τη σύνταξη και τις έννοιες της Ink, καθιστώντας την εκμάθησή της εύκολη, ακόμα και για χρήστες με μικρή εμπειρία στον προγραμματισμό. Ένα tutorial βασικών εννοιών και front-end customization μπορεί να βρεθεί στο επίσημο site ενώ το αναλυτικό εγχειρίδιο είναι διαθέσιμο στο επίσημο repository στο github, καθώς και στον Inky editor. Στο ίδιο repository υπάρχει επίσης εγχειρίδιο ειδικά για ενσωμάτωση Ink ιστοριών σε Unity projects μέσω της γλώσσας C#. Η παρούσα εργασία ανήκει σε αυτή την κατηγορία, επομένως το θέμα θα αναπτυχθεί αναλυτικά στη συνέχεια.

Βασικό tutorial: <https://www.inklestudios.com/ink/web-tutorial>

Αναλυτική εχειρίδιο:

<https://github.com/inkle/ink/blob/master/Documentation/WritingWithInk.md>

5.1.2.3 Ευέλικτο Integration

Η ομάδα της Inkle έχει δημιουργήσει plugin για το Unity engine, το οποίο επιτρέπει πλήρη ενσωμάτωση Ink αρχείων σε Unity projects. Πραγματοποιεί αυτόματα recompile στα αρχεία όσο γίνεται η επεξεργασία, επομένως το project ενημερώνεται αυτόματα. Παράλληλα, το game-development studio, *The Chinese Room*, έχει αναπτύξει ένα plugin που επιτρέπει ενσωμάτωση με projects του Unreal engine. Τα δυο αυτά plugins επιτρέπουν στους δημιουργούς που έχουν ένα ανάλογο προγραμματιστικό υπόβαθρο να χρησιμοποιήσουν την Ink ακόμα και για πιο μεγαλεπήβολα ανεξάρτητα παιχνίδια, πολλά εκ των οποίων αναπτύσσονται με τα δυο παραπάνω engines ως βασικά εργαλεία.

|  |
| --- |
| A screenshot of a computer  Description automatically generated  Εικόνα 8: Το interface τoυ Inky editor. |

5.2 Σύνταξη

5.2.1 Τα Βασικά Μέρη

* Περιεχόμενο

Εφόσον η Ink χρησιμοποιείται για συγγραφή αλληλεπιδραστικών μυθιστορημάτων, το βασικό περιεχόμενο ενός αρχείου (με την κατάληξη .ink) είναι απλό κείμενο (π.χ. “Hello World !”). Αλλάζοντας γραμμή, η Ink το απλώς αντιλαμβάνεται νέα παράγραφο.

Ως γλώσσα προγραμματισμού, σαφώς επιτρέπει στο χρήστη να εισάγει σχόλια που ο compiler δε θα λαμβάνει υπόψιν, εδώ με τη χρήση «//» για μια γραμμή, ή περικλείοντας μεγαλύτερο τμήμα κειμένου με τα σύμβολα «/\*» και «\*/» (Αν ο συγγραφέας θέλει να συμπεριλάβει κάποια υπενθύμιση για τον ίδιο, γράφοντας στην αρχή της γραμμής “TODO:”, ο compiler θα τυπώσει το ακόλουθο κείμενο.

Χρήσιμο εργαλείο αποτελούν οι ετικέτες (tags), διότι μπορούν να σηματοδοτήσουν σημεία του κειμένου, όπου ο δημιουργός επιθυμεί να προσθέσει επιπλέον περιεχόμενο όταν κάποια πλατφόρμα «τρέξει» το .ink αρχείο (πχ. στιλιστικές οδηγίες για την επεξεργασία του τρέχοντος κειμένου ή αναπαραγωγή πολυμέσων στο συγκεκριμένο σημείο). Οι ετικέτες δηλώνονται με το σύμβολο «#» (π.χ. Hello world ! #make\_it\_green).

* Επιλογές

Όπως έχει αναφερθεί, κύριο γνώρισμα της Ink είναι η εμφάνιση επιλογών στον παίκτη. Η δήλωση μιας επιλογής σηματοδοτείται με το σύμβολο «\*» στην αρχή. Το κείμενο που θα γράψει ο συγγραφέας κάτω από την επιλογή αποτελεί το κείμενο που θα εμφανιστεί εάν ο παίκτης διαλέξει τη συγκεκριμένη επιλογή. Ακολουθεί ένα παράδειγμα που ξεκαθαρίζει αρκετά σημεία:

Το ακόλουθο κείμενο:

"What's that?" my master asked.

\* "I am somewhat tired[."]," I repeated.

"Really," he responded. "How deleterious."

\* "Nothing, Monsieur!"[] I replied.

"Very good, then."

\* "I said, this journey is appalling[."] and I want no more of it."

"Ah," he replied, not unkindly. "I see you are feeling frustrated. Tomorrow, things will improve."

Παράγει το επόμενο αποτέλεσμα:

"What's that?" my master asked.

1: "I am somewhat tired."

2: "Nothing, Monsieur!"

3: "I said, this journey is appalling."

> 3

"I said, this journey is appalling and I want no more of it." "Ah," he replied, not unkindly. "I see you are feeling frustrated. Tomorrow, things will improve."

O δημιουργός θεωρητικά δύναται να προσθέσει όσες επιλογές θέλει, εδώ υπάρχουν τρεις. Ο παίκτης επέλεξε την τρίτη εναλλακτική και τυπώθηκε το αντίστοιχο κείμενο. Οι αγκύλες ( [ , ] ) μπορούν να χρησιμοποιηθούν για να σημάνουν ποια κομμάτια των επιλογών θα συμπεριληφθούν στη συνέχεια του κειμένου.

* Ομάδες (Knots)

Ενδεχομένως να γίνεται αντιληπτό ότι ορισμένες ορολογίες της Ink φαίνονται ιδιόμορφες και δε συναντώνται στην ελληνική βιβλιογραφία. Για να αποφευχθεί τυχόν σύγχυση, οι ορολογίες αυτές θα αναφέρονται με την πρωτότυπη αγγλική τους ορολογία.

Για να μπορεί η ιστορία να μεταβαίνει από το ένα σημείο στο άλλο μη γραμμικά , είναι απαραίτητο να οριοθετούνται και να προσδιορίζονται τα διάφορα τμήματα του κειμένου. Τα knots εξυπηρετούν αυτό το σκοπό, αποτελώντας τη δομική μονάδα της Ink. Μια καλή αναλογία θα ήταν να θεωρηθούν ως κεφάλαια ενός βιβλίου, τα οποία όμως ο αναγνώστης δεν επισκέπτεται κατ’ ανάγκη με τη σειρά, ενώ μπορεί να τα επισκεφθεί πάνω από μια φορά και κάθε φορά να παρατηρούνται αλλαγές. Υπό αυτή την έννοια, προσομοιάζουν τα δωμάτια των παραδοσιακών parsing games. Συμβολίζονται ως εξής:

=== back\_in\_london ===

We arrived into London at 9.45pm exactly.

* Εκτροπές (Diverts)

Οι εκτροπές εξυπηρετούν την ανάγκη του προγραμματιστή να μεταβιβάζει τη ροή της ιστορίας από ένα τμήμα της σε κάποιο άλλο, σε οποιαδήποτε γραμμή του αρχείου και αν αυτό βρίσκεται. Σηματοδοτείται με ένα βέλος (->), όπως παρακάτω :

=== back\_in\_london ===

We arrived into London at 9.45pm exactly.

-> hurry\_home

=== hurry\_home ===

We hurried home to Savile Row as fast as we could.

Η χρήση του βέλους υποδηλώνει τη μετάβαση σε νέα γραμμή (line-break), επομένως στην περίπτωση που είναι επιθυμητή η συνέχεια στην ίδια γραμμή, χρησιμοποιείται στο ενδιάμεσο το σύμβολο «<>». Τα παραπάνω εργαλεία αποτελούν τη βασική δομή για τη δημιουργία μιας απλής ιστορίας στην ink.

* Ραφές (Stitches)

Για πιο ξεκάθαρη και ευνόητη δόμηση της ιστορίας, ενθαρρύνεται η χρήση των stitches, τα οποία αποτελούν υποκατηγορία των knots. Συμβολίζονται με « = » και εξυπηρετούν όταν ο χρήστης επιθυμεί να οργανώσει το περιεχόμενο ενός knot, διαιρώντας το σε μικρότερα οριοθετημένα τμήματα για ευκολότερη αλληλεπίδραση μεταξύ τους, όπως βλέπουμε στη συνέχεια:

=== the\_orient\_express ===

We boarded the train, but where?

\* [First class] -> in\_first\_class

\* [Second class] -> in\_second\_class

= in\_first\_class

...

= in\_second\_class

...

* Μορφές Επιλογών

Η καθιερωμένη μορφή επιλογής (με το σύμβολο «\*») μπορεί να προσπελαστεί μόνο μία φορά. Πολλές φορές είναι χρήσιμο να υπάρχει κάποια δευτερεύουσα μορφή επιλογής, στην περίπτωση που οι διαθέσιμες επιλογές ενός τμήματος κειμένου εξαντληθούν, προκειμένου να μη δημιουργηθεί κενό στην ιστορία και τερματίσει η εφαρμογή πρόωρα. Αυτό επιτυγχάνεται με την εναλλακτική επιλογή (fallback choice), η οποία σηματοδοτείται ως «-> \*…».

Ένα διαφορετικό ενδεχόμενο είναι να χρειάζεται κάποια επιλογή που να μπορεί να είναι πάντοτε προσπελάσιμη. Μια τέτοια σηματοδοτείται με «+».

Η διαθεσιμότητα μιας επιλογής μπορεί να εξαρτάται και από παραμέτρους, συγκεκριμένα λογικές συνθήκες που σχηματίζονται από τις ενέργειες του παίκτη. Οι λογικές συνθήκες μπορούν να διατυπωθούν με τη βοήθεια των κλασικών λογικών τελεστών (and ή &&, οr ή ||, not ή !), ή ακόμα και απλά με μια ονομαστική δήλωση. Στο παρακάτω παράδειγμα φαίνονται μερικοί τρόποι με τους οποίους χρησιμοποιούνται λογικές συνθήκες για να επηρεάσουν την έκβαση της ιστορίας.

\* { not visit\_paris } [Go to Paris] -> visit\_paris

+ { visit\_paris } [Return to Paris] -> visit\_paris

\* { not (visit\_paris or visit\_rome) && (visit\_london || visit\_new\_york) } [Go where? I'm confused. ] -> visit\_someplace

Στην πραγματικότητα, ο ονομαστικός έλεγχος μιας σήμανσης εκφράζεται από έναν ακέραιο αριθμό που δηλώνει πόσες φορές ο παίκτης έχει επισκεφθεί ένα τμήμα κειμένου, όχι από τις λογικές τιμές true/false. Κατά αντιστοιχία, εάν ο παίκτης έχει επισκεφθεί, για παράδειγμα, ένα stitch τουλάχιστον μία φορά, η δήλωση αυτού επιφέρει αληθές λογικό αποτέλεσμα. Φυσικά, λόγω του αριθμητικού ελέγχου, ο προγραμματιστής μπορεί να συμπεριλάβει μια επιλογή του παρακάτω τύπου:

\* {seen\_clue > 3} [Flat-out arrest Mr Jefferson]

Υπάρχουν επίσης διάφοροι τρόποι για δημιουργία δυναμικού κειμένου, το οποίο διαφοροποιείται και αντικαθίσταται τμηματικά κάθε φορά που το συναντά ο παίκτης, προσδίδοντας μεγαλύτερη ζωτικότητα στην ιστορία. Τοποθετώντας διάφορες εκδοχές κειμένου μέσα σε άγκιστρα ορίζουμε ακολουθίες κειμένου και χρησιμοποιώντας ορισμένα σύμβολα (!,&,~) μπορούμε να καθορίσουμε αν αυτές οι ακολουθίες θα παρουσιάσουν τα περιεχόμενά τους μόνο μία φορά, κυκλικά ή με τυχαία φορά. Αναφορικά, δίνεται το κάτωθι παράδειγμα:

=== whack\_a\_mole ===

{I heft the hammer.|{~Missed!|Nothing!|No good. Where is he?|Ah-ha! Got him! -> END}}

The {&mole|{&nasty|blasted|foul} {&creature|rodent}} is {in here somewhere|hiding somewhere|still at large|laughing at me|still unwhacked|doomed}. <>

{!I'll show him!|But this time he won't escape!}

\* [{&Hit|Smash|Try} top-left] -> whack\_a\_mole

\* [{&Whallop|Splat|Whack} top-right] -> whack\_a\_mole

\* [{&Blast|Hammer} middle] -> whack\_a\_mole

\* [{&Clobber|Bosh} bottom-left] -> whack\_a\_mole

\* [{&Nail|Thump} bottom-right] -> whack\_a\_mole

\* [] Then you collapse from hunger. The mole has defeated you!

-> END

Επίσης, είναι εφικτό να τροποποιηθεί το κείμενο βάσει λογικών υποθέσεων. Στο επόμενο απόσπασμα, εάν η λογική πρόταση είναι αληθής, θα τυπωθεί η πρώτη εκδοχή του κειμένου, διαφορετικά θα τυπωθεί η δεύτερη:

"His real name was {met\_blofeld.learned\_his\_name: Franz|a secret}."

* Βασικές Συναρτήσεις

Η Ink διαθέτει άτυπα μια λίστα βασικών συναρτήσεων βιβλιοθήκης, οι οποίες διεκπεραιώνουν χρήσιμες απλές εργασίες.

Η συνάρτηση CHOICE\_COUNT( ) επιστρέφει τον αριθμό των διαθέσιμων επιλογών που έχει ο παίκτης στην παρούσα φάση. Στο ακόλουθο παράδειγμα, ο παίκτης έχει διαθέσιμες τις επιλογές B και C :

\* {false} Option A

\* {true} Option B

\* {CHOICE\_COUNT() == 1} Option C

Η συνάρτηση TURNS( ) επιστρέφει τον αριθμό των γύρων (ενεργειών του παίκτη) από την αρχή της ιστορίας. Επιπλέον, η συνάρτηση TURNS\_SINCE(->knot) επιστρέφει τον αριθμό των ενεργειών από την τελευταία φορά που ο παίκτης βρέθηκε στο συγκεκριμένο knot/stitch. Η τιμή 0 σημαίνει ότι ο παίκτης μόλις βρέθηκε εκεί (δεν έχει μεσολαβήσει ενέργειά του) και η τιμή -1 υποδηλώνει ότι ο παίκτης δεν έχει προσπελάσει το συγκεκριμένο σημείο.

* + 1. Weaves

Με τον όρο weave ονομάζεται η συντακτική δυνατότητα που παρέχει η ink για απλουστευμένη δόμηση της ιστορίας με ορισμένα εργαλεία που δεν έχουν αναφερθεί ακόμα.

* Συλλέκτες (Gathers)

Ένας συλλέκτης συμβολίζεται με « - » στην απλούστερη μορφή του και σηματοδοτεί ένα κοινό σημείο στο οποίο ο προγραμματιστής θέλει να καταλήγουν όλες οι επιλογές από το τρέχον σύνολο. Εφόσον, λοιπόν μπορεί να υπάρχει η επιθυμία ορισμένες ομάδες επιλογών να καταλήγουν συγκεντρωτικά στο ίδιο σημείο, εκεί δεν υπάρχει ανάγκη περιττών knots/stitches και diverts. Σε εμφωλευμένες περιπτώσεις, υφίστανται αντίστοιχα οι συμβολισμοί «--», «---» κλπ. Οι συλλέκτες δημιουργούν μια πιο ευανάγνωστη δομή, καθώς διασφαλίζουν τη ροή της ιστορίας από την κορυφή προς τα κάτω, περιορίζοντας αχρείαστες διακλαδώσεις και ρίσκο σφαλμάτων. Ακολουθεί ένα αντιπροσωπευτικό παράδειγμα :

"Well, Poirot? Murder or suicide?"

\* "Murder!"

"And who did it?"

\* \* "Detective-Inspector Japp!"

\* \* "Captain Hastings!"

\* \* "Myself!"

- - "You must be joking!"

\* \* "Mon ami, I am deadly serious."

\* \* "If only..."

\* "Suicide!"

"Really, Poirot? Are you quite sure?"

\* \* "Quite sure."

\* \* "It is perfectly obvious."

- Mrs. Christie lowered her manuscript a moment. The rest of the writing group sat, open-mouthed.

Εάν ο παίκτης επιλέξει την πρώτη επιλογή *(“Murder!”)*, υποβάλλεται μεταξύ τριών νέων επιλογών, όπου όλες, όμως, οδηγούν στην ίδια φράση *(“You must be joking!”)*. Κατόπιν, πρέπει να επιλέξει μεταξύ άλλων δύο επιλογών. Αν εξαρχής επιλέξει τη δεύτερη επιλογή, θα έχει να απλά διαλέξει μεταξύ δύο νέων επιλογών. Ωστόσο, όποια και αν είναι η πρώτη του κίνηση, εν τέλει θα καταλήξει στο τελευταίο gather *(Mrs. Christie lowered…)*.

* Ετικέτες (Labels)

Οι ετικέτες είναι ο ορισμός λέξεων μεταξύ παρενθέσεων σε κομβικά σημεία της ιστορίας, στα οποία μπορεί ο χρήστης αργότερα να αναφερθεί, να μεταβεί ή με βάση τα οποία να καθοριστεί η μετέπειτα πορεία. Αποτελούν έναν ευέλικτο τρόπο για τον προγραμματιστή να δημιουργεί περαιτέρω εξαρτήσεις της ιστορίας από προηγούμενα σημεία της. Μπορούν να χρησιμοποιηθούν τόσο σε gathers, όσο και σε επιλογές. Το επόμενο παράδειγμα είναι ιδιαίτερα περιεκτικό, καθώς αποτελεί έναν τυπικό διάλογο με ένα NPC (Non-Playable Character). Εδώ, ο παίκτης αρχικά μπορεί να διαλέξει μεταξύ τριών επιλογών. Και οι τρεις θα τον οδηγήσουν στο *(loop)*. Ύστερα, το πρώτο divert θα τον οδηγήσει και πάλι στο *(opts)*. Πλέον, δε θα διαθέτει την επιλογή που διάλεξε νωρίτερα, αλλά θα έχει τις δυο υπόλοιπες, καθώς και μια καινούρια, επειδή επισκέφθηκε το *(loop).* Τώρα, με τη νέα επιλογή, ο παίκτης μπορεί να μεταβεί κατευθείαν στο *(done)* ή να εξαντλήσει τις επιλογές του και να βρεθεί και πάλι εκεί, απλά με περισσότερα βήματα :

- (opts)

\* 'Can I get a uniform from somewhere?'[] you ask the cheerful guard. 'Sure. In the locker.' He grins. 'Don't think it'll fit you, though.' \* 'Tell me about the security system.' 'It's ancient,' the guard assures you. 'Old as coal.'

\* 'Are there dogs?' 'Hundreds,' the guard answers, with a toothy grin. 'Hungry devils, too.'

// We require the player to ask at least one question

\* {loop} [Enough talking] -> done

- (loop) // loop a few times before the guard gets bored

{ -> opts | -> opts | }

He scratches his head.

'Well, can't stand around talking all day,' he declares.

- (done)

You thank the guard, and move away.

Αναφορά σε κάποια ετικέτα μπορεί να γίνει και εκτός της εμβέλειας εντός της οποίας έχει οριστεί, χρησιμοποιώντας απλά ως πρόθεμα το όνομα του stitch/label στο οποίο την έχουμε ορίσει (π.χ. stitch\_one.label\_one).

5.2.3 Μεταβλητές και Λογική

H Ink υποστηρίζει επίσης τόσο καθολικές (global), όσο και προσωρινές (temporary) μεταβλητές ποικίλων τύπων δεδομένων. Η δήλωση τους είναι απλή και ο τύπος τους ορίζεται αυτόματα από τα δεδομένα που εισάγονται σε αυτές, όπως φαίνεται στη συνέχεια:

VAR knowledge\_of\_the\_cure = false

VAR players\_name = "Emilia"

VAR number\_of\_infected\_people =521

VAR current\_epilogue = -> they\_all\_die\_of\_the\_plague

Αξίζει να αναφερθεί ότι οι καθολικές μεταβλητές μπορούν να ανακτηθούν και να τροποποιηθούν άμεσα και σε runtime επίπεδο. Επίσης, παρέχουν ευκολία στην επικοινωνία μεταξύ του αρχείου της ιστορίας και του ευρύτερου project, καθώς με την αποθήκευση της κατάστασης της ιστορίας (story state), αποθηκεύεται κατά συνέπεια και η κατάσταση των μεταβλητών μέχρι το σημείο της ιστορίας κατά το οποίο έγινε η αποθήκευση, χωρίς να απαιτούνται επιπλέον ενέργειες.

* Λογική

Οι λογικές πράξεις και διεργασίες με μεταβλητές μπορούν να πραγματοποιηθούν παράλληλα με την εισαγωγή του κειμένου, χωρίς φυσικά να τυπώνονται, χρησιμοποιώντας το σύμβολο «~» ως πρόθεμα.

=== set\_some\_variables ===

~ knows\_about\_wager = true

~ x = (x \* x) - (y \* y) + c

~ y = 2 \* x \* y

Υποστηρίζονται κανονικά και οι βασικές μαθηματικές πράξεις (+ , - , \* , / , % ), η δύναμη ( POW( ) ), καθώς και οι συναρτήσεις RANDOM(min,max) INT( ), FLOOR( ) και FLOAT( ).

Ωστόσο, οι δυνατότητες που παρέχει για τη διαχείριση αλφαριθμητικών (strings) είναι περιορισμένες και αρκούνται στις βασικές διεργασίες ελέγχου ισότητας και περιεχομένου :

{ "Yes, please." == "Yes, please." }

{ "No, thank you." != "Yes, please." }

{ "Yes, please" ? "ease" }

* Υποθέσεις (if/else)

Ο έλεγχος υποθέσεων συναντάται κυρίως σε δυο γνώριμες μορφές, οι οποίες στην ουσία είναι οι if/else και switch:

{

- x == 0:

~ y = 0

- x > 0:

~ y = x – 1

- else:

~ y = x + 1

}

{ x:

- 0: zero

- 1: one

- 2: two

- else: lots

}

Η κάθε υπόθεση μπορεί επίσης να περιέχει απλό κείμενο, επιλογές και εκτροπές. Σημαντική σημείωση πως η προσθήκη επιλογών απαιτεί και χρήση αντίστοιχων εκτροπών, έτσι ώστε η ροή της ιστορίας να μπορέσει μετά να βγει εκτός του μπλοκ των υποθέσεων.

* Προσωρινές Μεταβλητές

Οι προσωρινές μεταβλητές αποσκοπούν σε γρήγορες ενέργειες και υπολογισμούς σε τοπικό επίπεδο και διαγράφονται όταν η ιστορία προσπερνάει την περιοχή εμβέλειάς τους, όπως εδώ:

=== near\_north\_pole ===

~ temp number\_of\_warm\_things = 0

{ blanket:

~ number\_of\_warm\_things++

}

{ ear\_muffs:

~ number\_of\_warm\_things++

}

{ gloves:

~ number\_of\_warm\_things++

}

{ number\_of\_warm\_things > 2:

Despite the snow, I felt incorrigibly snug.

- else: That night I was colder than I have ever been.

}

Μπορούν , όμως, επίσης να χρησιμοποιηθούν ως παράμετροι συναρτήσεων. Αυτή η ιδιότητα επιτρέπει τη χρήση προσωρινών μεταβλητών και εκτός των πεδίων στα οποία έχουν οριστεί, όπως γίνεται αντιληπτό στην επόμενη αναδρομική περίπτωση:

-> add\_one\_to\_one\_hundred(0, 1)

=== add\_one\_to\_one\_hundred(total, x) ===

~ total = total + x

{ x == 100: -> finished(total)

- else: -> add\_one\_to\_one\_hundred(total, x + 1)

}

=== finished(total) ===

"The result is {total}!" you announce. Gauss stares at you in horror. -> END

* Συναρτήσεις (Functions)

Στην Ink, οι συναρτήσεις ορίζονται ως knots, εισάγοντας ωστόσο ορισμένες ιδιομορφίες. Έχουν τη δυνατότητα να επιστρέφουν τιμές (όπως κάνουν γενικά οι τυπικές συναρτήσεις) και δεν επιτρέπουν στο σώμα τους τη εισαγωγή επιλογών, εκτροπών και ραφών. Παρατίθενται ορισμένα παραδείγματα συναρτήσεων που κατασκευάστηκαν στο πλάισιο της εργασίας :

|  |
| --- |
| A screenshot of a computer program  Description automatically generated  Εικόνα 9: Η συνάρτηση fight( ). |

Στην εικόνα [9] παρουσιάζεται η συνάρτηση *fight( ).* Η παραπάνω συνάρτηση λαμβάνει ως παράμετρο έναν ακέραιο, ο οποίος ορίζεται από τις επιλογές του *fight­­\_figure* (εικόνα [8]), όπου και καλείται η συνάρτηση. Με τη βοήθεια της RANDOM( ) δημιουργείται ένα σύστημα «τύχης», ώστε η ζημία που προκαλεί κάθε χτύπημα του χαρακτήρα και του αντιπάλου του να διακυμαίνεται ελαφρώς σε κάθε επανάληψη, προκειμένου η σκηνή της μάχης να γίνει πιο παραστατική και περιπετειώδης. Όσο η ένδειξη υγείας του χαρακτήρα ή του αντιπάλου είναι μεγαλύτερη του μηδενός, η συνάρτηση καλείται επαναληπτικά από το *fight­­\_figure*, μέσα από την επιλογή του παίκτη για το επόμενο χτύπημα. Όταν μία από τις παραπάνω ενδείξεις γίνει μικρότερη ή ίση του μηδενός, η ιστορία μεταβαίνει μέσω του *fight­­\_figure* στο ανάλογο stitch και συνεχίζει.

|  |
| --- |
| A screenshot of a computer  Description automatically generated  Εικόνα 10: Το stitch fight\_figure. |

Άλλη μια ενδιαφέρουσα συνάρτηση είναι η *try( ),* η οποία φαίνεται στην εικόνα [11] και αποτελεί βασικό σημείο αναφοράς του παιχνιδιού. Καλείται σε σημεία της ιστορίας όπου παρουσιάζεται στον παίκτη μια επιλογή, η οποία για να πραγματοποιηθεί επιτυχημένα, θα πρέπει η τιμή μιας ικανότητας του χαρακτήρα (υπαγόμενη στο *personality* ή *skill*) να υπερβεί ένα όριο, με τη επιπλέον βοήθεια της ρίψης ενός ζαριού που εκφράζεται από τη RANDOM( ). Το μονοπάτι της ιστορίας που θα ακολουθηθεί εξαρτάται από την έκβαση της επιλογής, δηλαδή αν θα πραγματοποιηθεί με επιτυχία ή όχι.

|  |
| --- |
| A computer screen shot of a code  Description automatically generated  Εικόνα 11: Η συνάρτηση try( ). |

Στο επόμενο παράδειγμα, παρατίθεται μια χρήση της συνάρτησης, όπου ο παίκτης αποφάσισε να παραβιάσει μια πόρτα. Εάν μια από τις ικανότητες *handyman* ή *stealthy* πληρούν την αντίστοιχη προϋπόθεση, ο χαρακτήρας καταφέρνει να μπει στο δωμάτιο. Σε διαφορετική περίπτωση, αναγκάζεται να καλέσει τον θυρωρό.

|  |
| --- |
| A screenshot of a computer code  Description automatically generated  Εικόνα 12: Κλήση της συνάρτησης try( ). |

Τέλος, μια ακόμη συνάρτηση που συμβάλλει στην προσομοίωση ενός world model είναι η *timepass( ).* Η συγκεκριμένη καλείται όταν πρέπει να αλλάξει το χρονικό πλαίσιο της ιστορίας, από μέρα σε νύχτα, ή το αντίστροφο. Ορισμένες επιλογές του παίκτη ή συμβάντα διαφοροποιούνται ανάλογα με την ώρα της μέρας. Στην εικόνα [14] παρουσιάζεται ένα στιγμιότυπο, όπου ορισμένες επιλογές είναι διαθέσιμες για συγκεκριμένες στιγμές της ημέρας.

|  |
| --- |
| A computer code with green and blue text  Description automatically generated  Εικόνα 13: Η συνάρτηση timepass( ). |

|  |
| --- |
| A computer screen shot of a computer code  Description automatically generated  Εικόνα 14: Στιγμιότυπο επιλογών που επηρεάζονται από την time\_of\_day  . |

5.2.4 Ροή Ελέγχου (Flow Control)

* Σήραγγες (Tunnels)

Η σύνταξη με σήραγγες επιτρέπει πιο ελεύθερη συγγραφή, καθώς ο συγγραφέας μπορεί με αυτή να εκτρέψει την ιστορία σε ένα διαφορετικό σημείο, αλλά μόλις το knot/stitch στο οποίο έγινε η εκτροπή φτάσει στο τέλος του, η ροή να συνεχίσει από το σημείο πριν την εκτροπή. Στην ακόλουθη περίπτωση φαίνεται η πρακτικότητα της συγκεκριμένης σύνταξης. Τα δυο τελευταία knots καλούν αμφότερα το πρώτο σε κάποιο σημείο του μπλοκ τους και μόλις αυτό φτάσει στο τέλος του, το κάθε knot συνεχίζει τη δική του ροή :

=== crossing\_the\_date\_line(-> return\_to) ===

...

* -> return\_to

...

=== outside\_honolulu ===

We arrived at the large island of Honolulu.

- (postscript)

-> crossing\_the\_date\_line(-> done)

- (done)

-> END

...

=== outside\_pitcairn\_island ===

The boat sailed along the water towards the tiny island.

- (postscript)

-> crossing\_the\_date\_line(-> done)

- (done)

-> END

Η σύνταξη του tunneling αποτελείται από δυο βέλη (->), τα οποία περικλείουν το τμήμα στο οποίο θα πραγματοποιηθεί η ενδιάμεση εκτροπή. Αυτό που αλλάζει στο ενδιάμεσο τμήμα είναι ότι στο τέλος του τοποθετούνται δύο βέλη (->->). Στο επόμενο παράδειγμα, γίνεται εκτροπή στο knot με όνομα *crossing\_the\_date\_line* και μόλις το περιεχόμενο αυτού φτάσει στο πέρας του, η ροή της ιστορίας συνεχίζει στην επόμενη εκτροπή :

-> crossing\_the\_date\_line ->

=== crossing\_the\_date\_line ===

//This is a tunnel! ...

* ->->
* Νήματα (Threads)

Αρχικά, είναι σημαντικό να καταστεί σαφές ότι δεν πρόκειται για πραγματικό threading, όπως αυτό ορίζεται στην επιστήμη των υπολογιστών. Στη ουσία, αποτελεί ένα τρόπο συνδυασμού κειμένου και επιλογών από διαφορετικά τμήματα της ιστορίας στο ίδιο σημείο της ροής της ιστορίας. Για να ορίσουμε ένα νήμα, τοποθετούμε ως πρόθεμα το ανάποδο βέλος (<-). Όταν ο παίκτης διαλέξει μια επιλογή από ένα νήμα, η ιστορία θα συνεχίσει με βάση αυτό και τα υπόλοιπα θα παραμείνουν απροσπέλαστα. Για αποφυγή σφαλμάτων και μη πεπερασμένων νημάτων, ενδείκνυται η σαφής δήλωση του σημείου όπου ολοκληρώνεται ένα νήμα με την ένδειξη ->DONE. Η χρήση νημάτων συνίσταται σε πιο σύνθετες ιστορίες με πολλά μεταβλητά μέρη. Για παράδειγμα, σε μια ιστορία που ο παίκτης μπορεί να εξερευνάει ελεύθερα ένα χάρτη, η ροή της ιστορίας όταν εισέρχεται σε ένα χώρο μπορεί να είναι η εξής :

CONST HALLWAY = 1

CONST OFFICE = 2

VAR player\_location = HALLWAY

VAR generals\_location = HALLWAY

VAR doctors\_location = OFFICE

== run\_player\_location

{

- player\_location == HALLWAY: -> hallway

}

== hallway ==

<- characters\_present(HALLWAY)

\* [Drawers] -> examine\_drawers

\* [Wardrobe] -> examine\_wardrobe

\* [Go to Office] -> go\_office

- -> run\_player\_location = examine\_drawers

// etc...

// Here's the thread, which mixes in dialogue for characters you share the room with at the moment.

== characters\_present(room)

{ generals\_location == room:

<- general\_conversation

}

{ doctors\_location == room:

<- doctor\_conversation

}

-> DONE

== general\_conversation

\* [Ask the General about the bloodied knife]

"It's a bad business, I can tell you."

- -> run\_player\_location

== doctor\_conversation

\* [Ask the Doctor about the bloodied knife]

"There's nothing strange about blood, is there?"

- -> run\_player\_location

Στην παρούσα διπλωματική εργασία εφαρμόστηκε μια απλούστερη προσέγγιση και δε χρησιμοποιήθηκαν σήραγγες και νήματα, αυτός είναι και ο λόγος που δεν γίνεται εκτενέστερη ανάλυση. Η αναφορά τους γίνεται περισσότερο σε εγκυκλοπαιδικό πλαίσιο.

5.2.5 Παρακολούθηση Κατάστασης (State Tracking)

Σε ιστορίες όπου υπάρχει έντονα το στοιχείο αλληλεπίδρασης και κάθε επιλογή μπορεί εν δυνάμει να τροποποιήσει πολλά στοιχεία και μεταβλητές, μεταβάλλοντας ριζικά τον κόσμο της ιστορίας. Παρόλο που η Ink δεν παρέχει ένα world modeling σύστημα, όπως οι γλώσσες που βασίζονται στο parsing, προσφέρει ένα ευέλικτο σύστημα παρακολούθησης καταστάσεων που αποτελεί μια αξιόπιστη εναλλακτική.

* Λίστες (Lists)

Η λίστα αποτελεί την κύρια δομή παρακολούθησης καταστάσεων. Ορίζεται με το πρόθεμα LIST και τα περιεχόμενά της χωρίζονται με κόμμα. Μπορεί να της ανατεθεί ύστερα κάποια τιμή που ανήκει στα περιεχόμενά της, δείχνοντας έτσι μια συγκεκριμένη κατάσταση :

LIST kettleState = cold, boiling, recently\_boiled

\* [Turn on kettle]

The kettle begins to bubble and boil.

~ kettleState = boiling

Εάν είναι επιθυμητό, μπορεί να της δοθεί εξαρχής μια τιμή κατά τον ορισμό της, τοποθετώντας τη συγκεκριμένη τιμή σε παρενθέσεις.

Οι καταστάσεις μιας λίστας μπορούν να χρησιμοποιηθούν και πέρα από τη λίστα στην οποία ορίστηκαν, αναθέτοντας τις τιμές της λίστας σε μεταβλητές. Έτσι, ένα state machine μπορεί να χρησιμοποιηθεί για πολλά αντικείμενα του κόσμου της ιστορίας :

LIST heatedWaterStates = cold, boiling, recently\_boiled

VAR kettleState = cold

VAR potState = cold

\* {kettleState == cold} [Turn on kettle]

The kettle begins to boil and bubble.

~ kettleState = boiling

\* {potState == cold} [Light stove]

The water in the pot begins to boil and bubble.

~ potState = boiling

Αξιοσημείωτο είναι πως οι λίστες της Ink δε θυμίζουν τις λίστες των υπόλοιπων γλωσσών. Εδώ, η λίστα, πέραν του ότι ορίζει τις καταστάσεις της ως μεταβλητές, ορίζεται και η ίδια ως μεταβλητή. Επομένως, θεωρητικά, μπορεί να της ανατεθεί ύστερα μια οποιαδήποτε τιμή (π.χ. ένας ακέραιος), ωστόσο κάτι τέτοιο είναι προτιμότερο να αποφεύγεται.

* Τιμές Λιστών

Στην Ink έχει καίρια σημασία η σειρά με την οποία ορίζουμε τις καταστάσεις μέσα σε μια λίστα. Στην ουσία, η λίστα αντιμετωπίζεται και ως απαριθμητής (enumerator), επομένως στις καταστάσεις ανατίθεται και ένας αύξων αριθμός, πράγμα που σημαίνει πως μπορούν να αντιμετωπισθούν ως αριθμοί και να υποβληθούν σε αριθμητικές συγκρίσεις και αυξομειώσεις :

LIST volumeLevel = off, quiet, medium, loud, deafening

VAR lecturersVolume = quiet

VAR murmurersVolume = quiet

{ lecturersVolume < deafening:

~ lecturersVolume++

{ lecturersVolume > murmurersVolume:

~ murmurersVolume++

The murmuring gets louder.

}

}

Εάν είναι επιθυμητό να χρησιμοποιηθεί η αριθμητική τιμή μιας κατάστασης, αυτή μπορεί να αντληθεί, βάζοντας το όνομα της κατάστασης μέσα στη συνάρτηση LIST\_VALUE( ).

* Λίστες Πολλών Τιμών

Οι λίστες και οι μεταβλητές που περιέχουν τιμές λιστών μπορούν να περιέχουν πάνω από μια μεταβλητή ταυτόχρονα. Μπορούν να θεωρηθούν και Boolean σύνολα. Στο επόμενο παράδειγμα, η λίστα *DoctorsInSurgery* περιέχει τρεις τιμές, χωρίς αυτό να σημαίνει ότι δε μπορεί αργότερα να προστεθεί ή να αφαιρεθεί κάποια από αυτές (Για παράδειγμα, ~ *DoctorsInSurgery += Denver* ή *~ DoctorsInSurgery -= Adams )*, αρκεί να έχει δηλωθεί μαζί με τη λίστα :

LIST DoctorsInSurgery = (Adams), Bernard, (Cartwright), Denver, Eamonn

Οι βασικοί λογικοί έλεγχοι για κενή λίστα, σύγκριση και συμπερίληψη τιμών είναι επίσης εφικτοί και βάσει των παραπάνω μπορούν να γίνουν εύκολα κατανοητοί ανατρέχοντας στο εγχειρίδιο της γλώσσας. Αξίζει όμως να γίνει μια συνοπτική αναφορά σε ορισμένες απλές συναρτήσεις.

Η LIST\_ALL( ) επιστρέφει όλα τα δηλωμένα στοιχεία μιας λίστας, ακόμη και αυτά που δεν είναι ενεργά στην τρέχουσα φάση της ιστορίας. Η LIST\_COUNT ( ) επιστρέφει τον αριθμό των περιεχομένων και οι LIST\_MIN( ) και LIST\_MAX( ) επιστρέφουν αντίστοιχα την πρώτη και την τελευταία τιμή. Επίσης, υπάρχουν οι LIST\_RANGE( ), που επιστρέφει ένα προκαθορισμένο τμήμα της λίστας και η LIST\_INVERT( ), η οποία αντιμεταθέτει τη λογική τιμή κάθε στοιχείου της λίστας :

LIST\_RANGE(list\_name, min\_value, max\_value)

=== function changingOfTheGuard

{!GuardsOnDuty:

// "is GuardsOnDuty empty right now?"

~ GuardsOnDuty = LIST\_ALL(Smith)

- else:

~ GuardsOnDuty = LIST\_INVERT(GuardsOnDuty)

}

5.2.6 Διεθνής υποστήριξη χαρακτήρων

Η Ιnk δεν φέρει περιορισμούς ως προς τη χρήση μη-ASCII χαρακτήρων μέσα στο απλό κείμενο του περιεχόμενου της ιστορίας. Ωστόσο, υπάρχει περιορισμός στους χαρακτήρες που μπορούν να χρησιμοποιηθούν για τα ονόματα των σταθερών, μεταβλητών, stitches, diverts και άλλων στοιχείων ροής (γνωστά και ως identifiers). Όμως υποστηρίζει αυτόματα μια λίστα προκαθορισμένων μη-ASCII συνόλων χαρακτήρων που μπορούν να χρησιμοποιηθούν ως identifiers. Γενικά, αυτά τα σύνολα έχουν επιλεγεί για να περιλαμβάνουν το αλφαριθμητικό υποσύνολο του επίσημου unicode εύρους χαρακτήρων, το οποίο θα επαρκούσε για την ονομασία των identifiers. Το εύρος που καλύπτουν αυτά τα σύνολα είναι διαθέσιμο στο εγχειρίδιο της γλώσσας.

6. Το περιβάλλον της Unity

Το game engine της Unity αποτελεί ένα από τα πιο διαδεδομένα εργαλεία ανάπτυξης εφαρμογών και παιχνιδιών, χάρη στη δωρεάν προσβασιμότητά του, την βολική διεπαφή για τον χρήστη, την υποστήριξη για πολλά λειτουργικά συστήματα, τα άμεσα και χρηστικά εργαλεία και γενικότερα για το εύρος δυνατοτήτων που παρέχει στους προγραμματιστές.

Ακολουθεί η επεξήγηση της φιλοσοφίας του και ορισμένων εννοιών και εργαλείων που χρησιμοποιήθηκαν για την υλοποίηση του παιχνιδιού.

6.1 Unity Editor

Ο Editor είναι τo κύριο περιβάλλον ανάπτυξης που παρέχει το game engine και περιλαμβάνει τα απαραίτητα εργαλεία. Αποτελείται από πολλά μέρη, τα κυριότερα των οποίων αναφέρονται στη συνέχεια.

|  |
| --- |
| A screenshot of a computer  Description automatically generated  Εικόνα 15: Τα βασικά μέρη του Editor (Πηγή: Unity Technologies). |

6.1.1 Scene View (Όψη σκηνής)

To Scene View αποτελεί το αλληλεπιδραστικό παράθυρο, στο οποίο ο προγραμματιστής μπορεί να επεξεργάζεται αντικείμενα του παιχνιδιού και να τα επιθεωρεί από διάφορες οπτικές. Παράλληλα, υπάρχει και ένα άλλο παράθυρο, το Game View, το οποίο συνιστάται για έλεγχο του παιχνιδιού κατά την εκτέλεση (playtest).

Είναι σημαντικό να διευκρινιστεί ότι ένα project μπορεί να διαθέτει πολλές σκηνές (Scenes) , όπου κάθε σκηνή παρουσιάζει διαφορετικό περιβάλλον ή χώρο του παιχνιδιού και αντίστοιχα έχει τα δικά της αντικείμενα. Ωστόσο, εδώ γίνεται λόγος για το παράθυρο που εξετάζει την κάθε σκηνή, το οποίο είναι ένα.

6.1.2 Hierarchy View (Ιεραρχία)

To Hierarchy View είναι ένα παράθυρο, στο οποίο παρατίθενται και οργανώνονται τα αντικείμενα (Game Objects) του project. Εκεί δημιουργούνται, καταστρέφονται και δέχονται επεξεργασία τα οποία εφμανίζονται στη σκηνή του παιχνιδιού. Είναι σημαντική η σειρά με την οποία τα αντικείμενα είναι τοποθετημένα στο Hierarchy View, καθώς ενδέχεται να τροποποιεί τη διάταξη με την οποία εμφανίζονται στη σκηνή. Επίσης, τοποθετώντας χειροκίνητα ένα αντικείμενο εντός ενός άλλου στο συγκεκριμένο παράθυρο, το Hierarchy View tο ορίζει ως παιδί του δεύτερου, με τις επακόλουθες συνέπειες της σχέσης γονέας – παιδί του αντικειμενοστραφούς προγραμματισμού. Για κάθε σκηνή που περιλαμβάνεται στο παιχνίδι, υπάρχει διαφορετικό τέτοιο παράθυρο.

|  |
| --- |
| A screenshot of a computer program  Description automatically generated  Εικόνα 16: Ενδεικτικό απόσπασμα του Hierarchy View. |

6.1.3 Project Window (Παράθυρο Εργασίας)

Το συγκεκριμένο παράθυρο περιέχει και διατηρεί οργανωμένα όλα τα αρχεία έχει που εισάγει ή δημιουργήσει ο προγραμματιστής για το project, στον φάκελο Assets. Αυτά μπορεί να είναι δομικά στοιχεία σκηνών, αντικείμενα, πολυμέσα ή αρχεία κώδικα (scripts), μεταξύ άλλων. Σε αντίθεση με το Hierarchy View που αλλάζει το περιεχόμενό του ανάλογα με τη σκηνή, το project window περιέχει τα πάντα. Από εκεί ο χρήστης μπορεί να σύρει με το ποντίκι τα διάφορα Assets, τόσο στο Scene, όσο και στο Hierarchy View.

6.1.4 Inspector Window (Παράθυρο Επιθεώρησης)

Εδώ, επιλέγοντας κάποιο αντικείμενο, παρουσιάζονται και τροποποιούνται οι ιδιότητές του (components). Μπορεί κάποιος να προσθέσει διάφορες ιδιότητες σε ένα αντικείμενο, είτε για αλλαγή στην παρουσία του, είτε στη λειτουργικότητά του. Για παράδειγμα, στην εικόνα [9] φαίνονται στο παράθυρο οι ιδιότητες του αντικειμένου Save Button. Το αντικείμενο αυτό προέρχεται από την κλάση Button και αναπαριστά ένα κουμπί, το οποίο πατάει ο παίκτης για να αποθηκεύσει την πρόοδό του.

Την ιδιότητα να εκτελεί μια ενέργεια εφόσον το πατήσει ο παίκτης, του την παρέχει το component με τον όνομα Button. Ωστόσο, για να αναγνωρίσει το σύστημα ποια ενέργεια να εκτελέσει, πρέπει να προστεθεί και ένα component με το αντίστοιχο script (SaveGameButton.cs). Έπειτα, το Button component περιέχει τη μέθοδο OnClick(), στην οποία μπορεί να γίνει η αντιστοίχιση του script που είναι υπεύθυνο για την ενέργεια της αποθήκευσης.

Σε αυτό το αντικείμενο έχουν προστεθεί μερικά ακόμα components, τα οποία συντονίζουν δευτερεύουσες διεργασίες. Το Hover Over είναι ένα ακόμη script που έχει συμπεριληφθεί ως component, δυο συναρτήσεις του οποίου χρησιμοποιούνται από το component με όνομα Event Trigger. Η μια (changeWhenHover()) αλλάζει χρώμα στο κουμπί όταν βρεθεί σε αυτό ο κέρσορας και η άλλη (changeWhenLeaves()) το επαναφέρει στο αρχικό του χρώμα όταν ο κέρσορας απομακρυνθεί.

|  |
| --- |
| A screenshot of a computer program  Description automatically generated  Εικόνα 17: Το παράθυρο Inspector για το αντικείμενο Save Button. |

6.2 Η Φιλοσοφία του Αντικειμενοστραφούς Προγραμματισμού

Ο Αντικειμενοστραφής Προγραμματισμός (Object-Oriented Programming) είναι ένα πρότυπο προγραμματισμού που οργανώνει τον κώδικα γύρω από αντικείμενα (objects) και όχι ενέργειες (actions). Στο game engine της Unity, o Αντικειμενοστραφής Προγραμματισμός παίζει καθοριστικό ρόλο στον τρόπο με τον οποίο οι προγραμματιστές δομούν τον κώδικά τους, επιτρέποντάς τους να δημιουργούν επαναχρησιμοποιήσιμα και αρθρωτά στοιχεία για την αποτελεσματική ανάπτυξη παιχνιδιών.

6.2.1 Αντικείμενα

Κάθε σκηνή σε ένα project αποτελείται εξ’ ολοκλήρου σχεδόν από αντικείμενα. Αυτά είναι στιγμιότυπα κλάσεων (instances), τις οποίες ορίζει ο προγραμματιστής μέσω κώδικα, σε αυτή την περίπτωση με scripts σε C#. Κάθε αντικείμενο αντιπροσωπεύει μια οντότητα, είτε πρόκειται για κάτι απτό εντός του πλαισίου του παιχνιδιού, όπως η φιγούρα ενός χαρακτήρα της ιστορία, είτε για κάτι που δεν έχει υπόσταση στον κόσμο του παιχνιδιού, αλλά εξυπηρετεί στη διαχείριση του παιχνιδιού. Η δεύτερη κατηγορία αφορά κυρίως στα κενά αντικείμενα (Empty Game Objects), στα οποία συχνά δίνονται ονόματα όπως GameManager και χρησιμοποιούνται για να συντονίζουν άλλα αντικείμενα εντός της σκηνής, να διαχειρίζονται τη λογική πίσω από το gameplay, να διαχειρίζονται γεγονότα και να ελέγχουν διάφορα εφέ ή animations, μεταξύ άλλων. Βοηθούν αρκετά στην διατήρηση τάξης σε μια σκηνή, ενώ επίσης δεν επιβαρύνουν αισθητά την απόδοση. Στην εικόνα [18] φαίνoνται αντικείμενα που ανήκουν στην πρώτη κατηγορία, αποτελούν ευρήματα που έχει συλλέξει ο χαρακτήρας του παίκτη. Στην εικόνα [19] παρουσιάζεται το Hierarchy View της σκηνής για εκείνη τη στιγμή. Όπως μπορεί να παρατηρηθεί, κάτω από το αντικείμενο Items υπάρχει ένα υποσύνολο αντικειμένων που αντιπροσωπεύουν τα ευρήματα. Ο λόγος που αρκετά από αυτά είναι σκιασμένα είναι επειδή στην προκειμένη φάση της ιστορίας τα συγκεκριμένα αντικείμενα δεν υπάρχουν στη συλλογή του παίκτη και είναι ανενεργά στη σκηνή, για αυτό και δεν εμφανίζονται.

|  |
| --- |
| A screenshot of a video game  Description automatically generated  Εικόνα 18: Στιγμιότυπο του παιχνιδιού από ευρήματα του χαρακτήρα (Inventory Panel). |

|  |
| --- |
| A screenshot of a computer  Description automatically generated  Εικόνα 19: To Hierarchy View τη στιγμή της εικόνας [18]. |

6.2.2 InkManager

Ένα παράδειγμα κενού αντικειμένου που συντονίζει το gameplay είναι το αντικείμενο με όνομα InkManager, το οποίο περιέχει script με κλάση ίδιου ονόματος. Αναλαμβάνει ρόλο κύριου συντονιστή, καθώς επιβλέπει τις παρακάτω ενέργειες:

* Ελέγχει τη ροή της ιστορίας, δηλαδή πότε μπορεί να συνεχίσει να παράγει κείμενο και πότε να εμφανίσει επιλογές.
* Δημιουργεί τα αντικείμενα κειμένου (κλάση TMP\_Text) και κουμπιών (κλάση Button) βάσει καθορισμένων προτύπων.
* Αναλαμβάνει τη φόρτωση των δεδομένων όταν ο παίκτης εκκινεί το παιχνίδι έχοντας ήδη αποθηκεύσει κάποια πρόοδο.
* Εντοπίζει μεταδεδομένα που εμπεριέχονται στο .ink αρχείο και προσαρμόζει τη σκηνή βάσει αυτών (Για παράδειγμα, συντονίζει τα ηχητικά εφέ και αλλάζει τη μουσική ανάλογα με το χώρο που βρίσκεται ο χαρακτήρας).

6.2.3 Κλάσεις

Τα αντικείμενα, όμως, πηγάζουν από κλάσεις (classes). Οι κλάσεις αποτελούν τα πρότυπα των αντικειμένων και τους μεταβιβάζουν τα δομικά τους χαρακτηριστικά, όπως τις ιδιότητες (properties) και τις μεθόδους (methods). Με τον όρο properties εννοούνται τύποι δεδομένων, όπως μεταβλητές και οι μέθοδοι είναι το ανάλογο των συναρτήσεων. Συγκεκριμένα, κάθε αντικείμενο που δημιουργείται βάσει μιας δεδομένης κλάσης διαθέτει τους ίδιους τύπους δεδομένων (properties) και τις ίδιες μεθόδους με την κλάση, όμως κάθε τέτοιο αντικείμενο διατηρεί τη δική του κατάσταση (state), δηλαδή μοναδικές τιμές των δεδομένων του και οι μέθοδοι αποκαλούνται behaviors.

Πρακτικά, η δημιουργία ενός αντικειμένου από μια κλάση γίνεται με μια διαδικασία που λέγεται instantiation. Στην απλούστερη μορφή του, μπορεί να γίνει στην C# με τον ακόλουθο τρόπο:

**Story \_story = new Story(\_inkJsonAsset.text);**

Στην προηγούμενη εντολή, η κλάση είναι η Story και το \_story είναι ένα αντικείμενο (ένα instance της Story). Υπάρχει και άλλος τρόπος να πραγματοποιηθεί instantiation και είναι ο ακόλουθος:

**TMP\_Text \_textField = Instantiate(narratorPrefab) as TMP\_Text;**

6.2.4 Prefabs

Αντίστοιχα, η κλάση είναι η TMP\_Text και το αντικείμενο είναι το \_textField. Το narratorPrefab εκφράζει μια έννοια στην οποία δεν έχει γίνει ακόμα αναφορά.

Prefab είναι ουσιαστικά ένα αντικείμενο, το οποίο αποτελεί πρότυπο για τα αντίγραφά του. Δηλαδή, αν ένα αντικείμενο μετατραπεί σε prefab, τότε για κάθε μετατροπή που πραγματοποιεί ο προγραμματιστής στα χαρακτηριστικά του, αυτή θα μεταφέρεται και στα αντίγραφα. Το ίδιο ισχύει και αν οι αλλαγές πραγματοποιηθούν σε οποιοδήποτε αντίγραφο. Αυτό εξυπηρετεί πολύ σε περιπτώσεις όπου σε ένα project πρέπει να παραχθούν πολλά αντίγραφα ενός αντικειμένου (για παράδειγμα, δέντρα για μια σκηνή δάσους), γιατί αν ο προγραμματιστής χρειαστεί να κάνει αλλαγές, μπορεί να τις κάνει όλες μαζί ταυτόχρονα αντί να πρέπει να τις επαναλάβει για το καθένα. Ωστόσο, δεν είναι απαραίτητο όλα τα αντίγραφα ενός prefab να είναι ακριβώς ίδια. Υπάρχει η δυνατότητα ατομικών παραλλαγών, αλλά και ομαδικών (variants). Η δημιουργία ενός prefab γίνεται απλά όταν μεταφερθεί χειροκίνητα το αντικείμενο στο παράθυρο εργασίας (project window).

Από το παραπάνω παράδειγμα κώδικα, πηγάζει μια ακόμη χρησιμότητα των prefabs. Επιτρέπουν τη δημιουργία αντικειμένων βάσει του προτύπου τους κατά το χρόνο εκτέλεσης (runtime), μέσα από το script. Έτσι, λοιπόν, η παραπάνω γραμμή κώδικα δημιουργεί ένα αντικείμενο πεδίου κειμένου βάσει της κλάσης TMP\_Text με τα χαρακτηριστικά ενός συγκεκριμένου prefab.

Στην εικόνα [20] φαίνεται ένα στιγμιότυπο του παιχνιδιού. Επειδή το κείμενο της αφήγησης και τα κουμπιά επιλογών πρέπει να ξαναδημιουργούνται συχνά, κάτι τέτοιο είναι εφικτό μέσω των prefabs :

|  |
| --- |
| A screenshot of a video game  Description automatically generated  Εικόνα 20: Prefabs κειμένου και κουμπιών επιλογής. |

6.2.5 Scriptable Objects

Μια επιπλέον χρήσιμη κλάση στην ανάπτυξη παιχνιδιών είναι η κλάση ScriptableObject. Η ScriptableObject επιτρέπει τη δημιουργία δομών δεδομένων, οι οποίες μπορούν να χρησιμοποιηθούν σε όμοια αντικείμενα, εξατομικεύοντάς τα ή να αποθηκεύσουν μια συλλογή κοινών δεδομένων που προορίζονται για πολλά αντικείμενα διαφόρων σκηνών, χωρίς να χρειαστεί να δημιουργηθούν αντίγραφα των δεδομένων. Σε αντίθεση με τη MonoBehaviour, τα αντικείμενα που δημιουργούνται από την ScriptapleObject θεωρούνται assets, συνεπώς είναι διαρκώς διαθέσιμα σε κάθε σκηνή. Yπάρχει μόνο μια φορά στη μνήμη και αποφεύγεται η σπατάλη επανάληψης περιττών δεδομένων. Επιπλέον, είναι κλάση που ασχολείται καθαρά με αρχιτεκτονική δεδομένων και όχι λογικές εργασίες, επομένως συμβάλλει σε ευκολότερη συντήρηση κώδικα.

Όσον αφορά στην παρούσα εργασία, η ScriptableObject χρησιμοποιήθηκε για τη δημιουργία ενός πακέτου δεδομένων, τα οποία χρησιμοποιούνται για καρτέλες χαρακτήρων της ιστορίας. Ουσιαστικά, ορίζοντας μόνο μια κλάση από ScriptableObject, μπορεί να δημιουργηθεί ένα asset για κάθε χαρακτήρα και ορίζεται και μια από MonoBehaviour για να περαστούν τα δεδομένα του κάθε asset στο εκάστοτε αντικείμενο. Τα δεδομένα που χρειάζονται για κάθε χαρακτήρα είναι ένα αλφαριθμητικό για το όνομά του, άλλο ένα για την περιγραφή του και ένα αντικείμενο κλάσης Sprite για την φωτογραφία του. Έχοντας συμπεριλάβει το script της εικόνας [21] στα Assets, είναι δυνατό να δημιουργηθούν αντικείμενα του τύπου Character Profile (εικόνα [22]). Οι φωτογραφίες μεταφέρονται στο αντίστοιχο Sprite πλαίσιο των νέων Assets από το Inspector Window, ενώ τα αλφαριθμητικά πληκτρολογούνται επί τόπου στο Inspector Window. Απομένει η προσθήκη των prefabs για το σώμα της καρτέλας των χαρακτήρων στη σκηνή. Τοποθετώντας τα prefabs, προστίθενται σε αυτά components του script της CreateProfile (εικόνα [23]), που αντιστοιχίζει τα δεδομένα του κάθε Character Asset με τα ανάλογα children objects της καρτέλας.

|  |
| --- |
| A screen shot of a computer program  Description automatically generated  Εικόνα 21: Η κλάση CharacterProfile που προέρχεται από την ScriptableObject. |

|  |
| --- |
| A screenshot of a computer  Description automatically generated  Εικόνα 22: Δημιουργία Character Asset από το Project Window. |

|  |
| --- |
| A screenshot of a computer program  Description automatically generated  Εικόνα 23: Η κλάση CreateProfile. |

|  |
| --- |
| A screenshot of a video game  Description automatically generated  Εικόνα 24: Οι καρτέλες των χαρακτήρων. |

6.2.6 Πολυμορφισμός

Μια ακόμη καίρια ιδιότητα των κλάσεων είναι ο πολυμορφισμός (polymorphism). Ο πολυμορφισμός χαρακτηρίζεται από την ικανότητα μιας κλάσης να κληρονομεί (inherit) από άλλες κλάσεις, δημιουργώντας μια ιεραρχική σχέση. Αυτό σημαίνει πως ένα αντικείμενο που πηγάζει από μια child κλάση, διαθέτει και τα properties και methods της parent κλάσης και συνεπώς, πολλά αντικείμενα διαφορετικού τύπου μπορούν να εκτελούν τις ίδιες εργασίες που τους επιτρέπει η κοινή κλάση βάσης (base class), καθιστώντας αφενός απλούστερη τη συντήρηση και ανάπτυξη του κώδικα και δημιουργώντας αφετέρου πιο ευέλικτα και εξειδικευμένα αντικείμενα.

7. Ενσωμάτωση στη Unity

Όπως αναφέρθηκε νωρίτερα, η ομάδα της Inkle έχει δημιουργήσει ένα πακέτο, το οποίο επιτρέπει την ενσωμάτωση των .ink αρχείων σε ευρύτερα projects της Unity. Στην παρούσα υποενότητα θα παρουσιαστεί η προεργασία της ενσωμάτωσης και οι βασικές λειτουργίες που αυτή επιτρέπει μέσα από τη γλώσσα της C#. Τα αποσπάσματα κώδικα που θα ακολουθήσουν αφορούν στον κώδικα που συντάχθηκε για το παιχνίδι *Beyond Man*, το οποίο αποτελεί και αντικείμενο της εργασίας.

7.1 Προεργασία

Αρχικά, εφόσον κατέβει το αρχείο του πακέτου ενσωμάτωσης ( το οποίο μπορεί να βρεθεί στο σύνδεσμο:<https://github.com/inkle/ink/blob/master/Documentation/RunningYourInk.md>), πρέπει να τοποθετηθεί εντός του Unity project, στο φάκελο Assets. Ύστερα, εφόσον έχει συμπεριληφθεί και το .ink αρχείο της ιστορίας, ο editor της Unity επιτρέπει τη δοκιμή του παιχνιδιού (σαν προεπισκόπιση).

|  |
| --- |
| A screenshot of a video game  Description automatically generated  Εικόνα 25: Ο φάκελος του πακέτου της Ink, εντός του φακέλου Assets. |

Ουσιαστικά, χρησιμοποιείται ένα νέο αρχείο σε μορφή .json, το οποίο έχει μεταγλωττιστεί από τα αρχικά αρχεία .ink. Το πακέτο ενσωμάτωσης μεταγλωττίζει αυτόματα τα αρχεία Ink για τον χρήστη, ενώ υπάρχει επίσης η δυνατότητα μεταγλώττισης μέσω της γραμμής εντολών. Εντός του φακέλου Assets βρίσκονται και τα .cs scripts (δηλαδή γραμμένα σε C#) , τα οποία διαχειρίζονται τη ροή και τη λειτουργία του συνολικού παιχνιδιού.

Απαραίτητο εδώ είναι το αρχείο της ιστορίας να οριστεί στην κλάση MonoBehaviour, η οποία είναι και αυτή από την οποία όλα τα Unity scripts προέρχονται αρχικά. Η συγκεκριμένη κλάση εξυπηρετεί για τους παρακάτω λόγους:

* Μέθοδοι όπως οι Start(), Update(), FixedUpdate(), LateUpdate(), OnCollisionEnter(), ο οποίες είναι απαραίτητες για να διαχειρίζονται τα events της εφαρμογής, προέρχονται από την Monobehaviour. Οι συγκεκριμένες ανήκουν στον βρόχο παιχνιδιού του Unity και καλούνται αυτόματα από το Unity σε καθορισμένες χρονικές στιγμές.
* H MonoBehaviour παρέχει σημαντικά components, όπως τα GetComponent<T>(), Instantiate() και Destroy(), τα οποία επιτρέπουν τη διαχείριση των αντικειμένων που δημιουργεί ο προγραμματιστής στη σκηνή (Unity Scene). Γενικά, παρέχει πολλές μεθόδους όταν ο προγραμματιστής χρειάζεται να συνδέσει ένα script με ένα αντικείμενο της σκηνής (GameObject) για να μπορεί να το διαχειριστεί.
* Ένα πολύ βασικό component που παρέχει η MonoBehaviour είναι το Coroutine. Αυτό επιτρέπει τη διακοπή της εκτέλεσης μιας συνάρτησης και τη συνέχιση αυτής αργότερα (π.χ. μετά από αναμονή μερικών δευτερολέπτων). Αποτελεί ένα ουσιαστικό εργαλείο για τη διαχείριση χρονικών συμβάντων ή διαδοχικών λειτουργιών. Στην προκειμένη περίπτωση, η μέθοδος StartCoroutine θα εμφανίσει σταδιακά το κείμενο της ιστορίας, δημιουργώντας ένα οπτικό εφέ πληκτρολόγησης, και μόνο όταν αυτό ολοκληρωθεί θα εκτελέσει την DisplayButtons().

|  |
| --- |
| Εικόνα 26: Η μέθοδος StartCoroutine. |

* Μέσω της MonoBehaviour, oι καθολικές μεταβλητές έχουν το πλεονέκτημα να είναι προσβάσιμες και από στο Unity Inspector, διευκολύνοντας την τροποποίηση των τιμών απευθείας από το API (Application Programming Interface) της Unity . Αυτό διευκολύνει την προσαρμογή των ρυθμίσεων, χωρίς να χρειάζεται τροποποίηση του κώδικα.

7.2 Χρόνος Εκτέλεσης και API

7.2.1 Μετατροπή .ink σε .json

Όπως αναφέρθηκε παραπάνω, τα .ink αρχεία μεταγλωττίζονται σε αρχεία .json. Η Unity αναγνωρίζει το συγκεκριμένο με την κλάση TextAsset. Αυτό θα χρησιμοποιηθεί από ένα αντικείμενο της κλάσης Story, και έτσι η κατάσταση της ιστορίας θα είναι πλήρως διαχειρίσιμη μέσω του script. Πλέον, ο χρήστης θα καλεί την ιστορία σε επαναληπτικούς βρόχους. Στην επόμενη φωτογραφία φαίνεται ο κώδικας της εκκίνησης της εφαρμογής, όπου πραγματοποιείται η ανάθεση του .json αρχείου στη μεταβλητή \_story, καθώς και ο έλεγχος αν η εφαρμογή εκκινεί ύστερα από αποθήκευση, προκειμένου να φορτωθεί η ιστορία από ανάλογο σημείο, πράγμα που θα αναλυθεί στην πορεία.

|  |
| --- |
| A computer screen shot of a program code  Description automatically generated  Εικόνα 27: Κώδικας εκκίνησης της εφαρμογής. |

7.2.2 Ροή της ιστορίας

Κομβικό ρόλο έχει η Boolean τιμή canContinue της κλάσης Story, η οποία καλείται τακτικά για να εξετάσει εάν η ιστορία μπορεί να συνεχίσει. Συγκεκριμένα, επιστρέφει αληθή τιμή, όσο υπάρχει ακόμα κείμενο που μπορεί να τυπωθεί. Στην περίπτωση που ακολουθούν επιλογές ή η ιστορία δεν μπορεί να συνεχίσει (είτε γιατί έφτασε στο τέλος, είτε γιατί υπήρξε κάποιο σφάλμα), επιστρέφει false. Με τη μέθοδο Continue(), η ιστορία μεταβαίνει στην επόμενη γραμμή κειμένου, ενώ με την ContinueMaximally() μεταβαίνει στο σημείο που ο παίκτης καλείται να δει τις επόμενες επιλογές της ιστορίας. Η διαφορά είναι πως η Continue() πρέπει να καλείται εντός επαναληπτικού βρόχου για να πετύχει το ίδιο αποτέλεσμα :

**while (\_inkStory.canContinue) {**

**Debug.Log (\_inkStory.Continue ());**

**}**

Παρόλα αυτά, στο πλαίσιο αυτής της διπλωματικής υιοθετήθηκε μια διαφορετική προσέγγιση. Στην εικόνα [28], η Continue() καλείται σε πρώτη φάση εντός της συνάρτησης DisplayNextLines(). Στη εικόνα [29], καλείται σε δεύτερη φάση εντός της συνάρτησης WriteMoreLines(). Ουσιαστικά, στο παιχνίδι, ο παίκτης βλέπει το κείμενο να εμφανίζεται διαδοχικά σε ζευγάρια, πρώτα μέσω της DisplayNextLines() και έπειτα μέσω της WriteMoreLines(). Αφού εμφανιστεί το κάθε ζευγάρι, αν υπάρχουν διαθέσιμες επιλογές, αυτές εμφανίζονται, διαφορετικά εμφανίζεται ένα κουμπί, το οποίο πατάει ο παίκτης για να συνεχιστεί η ιστορία. Και στις δύο περιπτώσεις, η διαδικασία επαναλαμβάνεται.

|  |
| --- |
| A computer screen shot of a program code  Description automatically generated  Εικόνα 28: Κλήση της Continue() από την DisplayNextLines(). |
| A computer screen shot of a program code  Description automatically generated  Εικόνα 29: Κλήση της Continue() από την WriteMoreLines(). |

Όταν η canContinue επιστρέφει ψευδή τιμή, πρέπει να ελεγχθεί αν υπάρχουν επιλογές διαθέσιμες για τον παίκτη. Στην προκειμένη περίπτωση, η συνάρτηση DisplayChoices() ελέγχει πρώτα αν υπάρχουν ήδη επιλογές, ύστερα εάν τερμάτισε αναμενόμενα η ιστορία για να επιστρέψει στο αρχικό μενού του παιχνιδιού και εφόσον δεν ισχύει τίποτα από τα παραπάνω, αρχίζει ένα βρόχο για να συλλέξει τις διαθέσιμες επιλογές από το αρχείο της ιστορίας και να δημιουργήσει κουμπιά (Buttons) για τις επιλογές:

|  |
| --- |
| Εικόνα 30: Η συνάρτηση DisplayChoices(). |

7.2.3 Αποθήκευση - Φόρτωση

Για την αποθήκευση και τη φόρτωση της ιστορίας χρησιμοποιούνται αντίστοιχα οι μέθοδοι \_inkStory.state.ToJson() και \_inkStory.state.LoadJson(). Η πρώτη επιστρέφει μεταβλητή, και συγκεκριμένα τύπου string, εφόσον χειρίζεται αρχείο .json. Στη συγκεκριμένη εργασία, δημιουργήθηκε μια δομή αποθήκευσης που εκτός της κατάστασης της ιστορίας συμπεριλαμβάνει και τα περιεχόμενα διαφόρων αντικειμένων που κοσμούν τη βασική σκηνή του παιχνιδιού, όπως φόντο κειμένου, φόντο πλαισίου και μουσικό θέμα. Στην εικόνα [31], χρησιμοποιώντας μια αυτοσχέδια κλάση με όνομα SaveData, η συνάρτηση CreateSaveGameObject() συγκεντρώνει τις τελευταίες τιμές των αντικειμένων της σκηνής και τις επιστρέφει εντός της συνάρτησης SaveGame() ως μια νέα δομή SaveData. Ύστερα, η SaveGame() μετατρέπει τα δεδομένα σε δυαδική μορφή και τα σειριοποιεί, αποθηκεύοντάς τα τελικά σε ένα αρχείο, η διαδρομή του οποίου είναι σταθερή και παρέχεται από τη Unity. Ανάλογα, στην εικόνα [32], η συνάρτηση LoadGame() λαμβάνει το μονοπάτι στο οποίο βρίσκεται το αρχείο με τις αποθηκευμένες μεταβλητές και με την αντίστροφη διαδικασία ανακτά τις μεταβλητές. Έπειτα, αντιστοιχεί τις τιμές σε μεταβλητές της συνάρτησης LoadState του βασικού script και επιστρέφει στη συνάρτηση StartGame(), όπου εκκινεί το παιχνίδι.

|  |
| --- |
| A screenshot of a computer program  Description automatically generated  Εικόνα 31: Οι συναρτήσεις CreateSaveGameObject() και SaveGame(). |
| A computer screen with text and images  Description automatically generated  Εικόνα 32: Η συνάρτηση LoadGame(). |

Η συνάρτηση LoadState που ανήκει στο script με όνομα InkManager φαίνεται παρακάτω:

|  |
| --- |
| A screen shot of a computer code  Description automatically generated  Εικόνα 33: Η συνάρτηση LoadState(). |

Συχνά μπορεί να γίνουν λάθη στην Ink, τα οποία ο compiler δε μπορεί να εντοπίσει. Για να αποφευχθούν ανεξήγητα σφάλματα, ενδείκνυται η συμπερίληψη ενός διαχειριστή σφαλμάτων στο τμήμα του κώδικα που αφορά την εκκίνηση της εφαρμογής, όπως παρακάτω:

|  |
| --- |
| A computer screen shot of text  Description automatically generated  Εικόνα 34: Διαχειριστής σφαλμάτων. |

7.3 Ετικέτες (Tags)

Ένα ακόμη χρήσιμο εργαλείο της Ink είναι τα tags (για αποφυγή σύγχυσης με τα labels δε θα αναφέρονται ως ετικέτες). Συνιστούν έναν ευέλικτο και μη παρεμβατικό μηχανισμό που χρησιμοποιείται προς επισήμανση συγκεκριμένων τμημάτων της ιστορίας, χωρίς να επηρεάζεται η ροή ή η λογική της. Οι ετικέτες χρησιμοποιούνται κυρίως για σήμανση μεταδεδομένων (metadata), δηλαδή σύντομες λέξεις που μεταβιβάζονται σε άλλα τμήματα της εφαρμογής και υποδηλώνουν πληροφορία που πρέπει να παραχθεί ή να τροποποιηθεί.

Οι ετικέτες δεν αναλύονται ούτε μεταγλωττίζονται από το χρόνο εκτέλεσης στο πλαίσιο της διαδικασίας λήψης αποφάσεων της ιστορίας. Στην περίπτωση της παρούσας εργασίας, οι ετικέτες τοποθετούνται στο .ink αρχείο για να δηλώσουν στο API της Unity σημεία της ιστορίας όπου αλλάζει το φόντο του κειμένου, ανάλογα με το αν υπάρχει τριτοπρόσωπη αφήγηση, εσωτερικός μονόλογος ή διάλογος και σημεία όπου αλλάζει το σκηνικό και η μουσική ή τα ηχητικά εφέ, ανάλογα με το χώρο και την κατάσταση που βρίσκεται ο πρωταγωνιστής.

Μπορούν να τοποθετηθούν σε οποιοδήποτε σημείο της ιστορίας του .ink αρχείου. Συγκεκριμένα, κάθε φορά που καλείται η Continue(), τα tags εκείνης της γραμμής είναι διαθέσιμα για ανάκτηση με τη μέθοδο inkStory.currentTags, η οποία επιστρέφει μια λίστα αλφαριθμητικών στη C#. Η μέθοδος θα συλλέξει όλα τα tags που αντιστοιχούν στη γραμμή, ανεξάρτητα αν βρίσκονται μπροστά, πίσω ή ακριβώς από πάνω. Σε ορισμένες περιπτώσεις, μια χρήσιμη πρακτική είναι η τοποθέτηση κάποιων tags στην κορυφή ενός knot, διότι τότε τα συγκεκριμένα tags είναι διαθέσιμα από το API ανά πάσα στιγμή, χωρίς να χρειαστεί να φτάσει η ροή της ιστορίας σε αυτό το σημείο. Αν βρίσκονται tags στην κορυφή του αρχείου, είναι και αυτά πάντοτε διαθέσιμα με τη μέθοδο globalTags.

Στο ακόλουθο παράδειγμα υπάρχει ένα stitch, το οποίο στην κορυφή του περιέχει δυο tags που δηλώνουν την αλλαγή σκηνικού και λίγο παρακάτω ένα ακόμη που δηλώνει την αλλαγή μουσικής. Μια αναδρομή στην εικόνα [28] δείχνει πως η συνάρτηση Adjust() βρίσκεται εντός της DisplayNextLines() και καλείται αν η μέθοδος tags.Contains(“backgrounds”) κριθεί αληθής, που στην προκειμένη περίπτωση ισχύει. Στην εικόνα [36] παρουσιάζεται η συνάρτηση Adjust(), η οποία αναθέτει στο component του σκηνικού την αντίστοιχη εικόνα, ελέγχοντας το αλφαριθμητικό του tag. Ο πίνακας εικόνων ImagesArray[ ] έχει οριστεί στο script και έχει τοποθετηθεί ως component στο Unity Inspector, επιτρέποντας την άμεση χειροκίνητη τοποθέτηση εικόνων στον πίνακα μέσα από το φάκελο των Assets. Καθ’ αυτόν τον τρόπο, η συνάρτηση Adjust() ψάχνει μέσω if/else σε ποια θέση του ImagesArray αντιστοιχεί η εικόνα που σχετίζεται με το αλφαριθμητικό “workshop” για να την αναθέσει στο spriteRenderer.

|  |
| --- |
| A close-up of a white background  Description automatically generated  Εικόνα 35: Χρήση tags στην Ink. |
| A computer screen shot of a program code  Description automatically generated  Εικόνα 36: Η συνάρτηση Adjust(). |

7.4 Μεταβάσεις μέσω API

Η Unity επιτρέπει τη μετάβαση στα διάφορα σημεία της ιστορίας και μέσω των δικών της scripts. Συγκεκριμένα, η μέθοδος \_inkStory.ChoosePathString("myKnotName") δίνει τη δυνατότητα στο engine να μεταβεί στο knot με το συγκεκριμένο όνομα και έπειτα με κλήση της Continue(), η ιστορία θα συνεχίσει από εκεί, ανεξαρτήτως σημείου που βρισκόταν πριν. Το ίδιο ισχύει και για μετάβαση σε stitch, με τη διαφορά ότι πρέπει να δηλωθεί συγκεκριμένα, δηλαδή \_inkStory.ChoosePathString("myKnotName.theStitchWithin").

Σε κάποιο σημείο της ιστορίας του Beyond Man, o παίκτης καλείται να χρησιμοποιήσει το πληκτρολόγιο για να εισάγει έναν κωδικό. Κάτι τέτοιο δε συνηθίζεται στα choice-based παιχνίδια, επομένως η Ink δεν παρέχει άμεση υποστήριξη για είσοδο από το πληκτρολόγιο. Ωστόσο, με τη βοήθεια του κατάλληλου script στο API, η ChoosePathString μπορεί να επαναφέρει τον παίκτη στη ροή του παιχνδιού. Σε αμφότερες τις συναρτήσεις DisplayNextLines() και WriteMoreLines() περιέχεται η συνάρτηση CheckForInput(). Η συνάρτηση αυτή ελέγχει την τρέχουσα γραμμή της Ink για τυχόν tag με όνομα “Input” και αν το συναντήσει ενεργοποιεί στη σκηνή του παιχνιδιού ένα πεδίο εισαγωγής κειμένου με τη μέθοδο της εγχώριας κλάσης GameObject, SetActive(). Παράλληλα, έχει οριστεί στη σκηνή ένα GameObject με όνομα InputField, που αναπαριστά το πεδίο στο οποίο θα πληκτρολογήσει ο παίκτης. Στο εν λόγω αντικείμενο, έχει προστεθεί μέσω του Inspector το component onEndEdit και σε αυτό, το script που περιέχει τη συνάρτηση ReadString(). Το onEndEdit καλεί τη συνάρτηση ReadString() όταν ο παίκτης γράψει τον κωδικό και πατήσει τη λέξη. Εάν ο κωδικός είναι ο σωστός σύμφωνα με την ιστορία, η ReadString() καλεί με τη σειρά της την ChoosePathString, η οποία μεταφέρει την ιστορία στο αντίστοιχο stitch και η ροή του .ink αρχείου συνεχίζει κανονικά :

|  |
| --- |
| A computer screen with text  Description automatically generated  Εικόνα 37: Η συνάρτηση CheckForInput. |
| A computer screen shot of a program code  Description automatically generated  Εικόνα 38: Η συνάρτηση ReadString(). |

7.5 Αλληλεπίδραση με Μεταβλητές και Λίστες

7.5.1 Η συνάρτηση DisplayAttributes()

Μια κοινή πρακτική σε Unity projects που ενσωματώνουν το πακέτο της Ink είναι η πρόσβαση λιστών και μεταβλητών του αρχείου της ιστορίας από το Unity API. Εξυπηρετεί όταν ο δημιουργός θέλει να εμφανίζει τιμές και στοιχεία από αυτά τα δεδομένα στη σκηνή του παιχνιδιού ή να τα τροποποιεί μέσω άλλων διαδικασιών από τον κώδικα της C#. Για παράδειγμα, όταν υπάρχει η επιθυμία να εμφανίζονται στον παίκτη στοιχεία του χαρακτήρα του σε ένα γραφικό περιβάλλον και όχι απλά τυπωμένα σε κείμενο, στοιχεία όπως η υγεία ή τα αντικείμενα που έχει συλλέξει. ‘Άλλη περίπτωση είναι να επιτρέπεται στον παίκτη να τροποποιεί της δεξιότητες του χαρακτήρα του μέσω ενός γραφικού πανελ, στο οποίο το αρχείο της ιστορίας δεν έχει πρόσβαση.

Στην εικόνα [39] παρουσιάζεται απόσπασμα της συνάρτησης DisplayAttributes(). Με χρήση της μεθόδου variablesState[], συλλέγονται από το αρχείο της ιστορίας οι τιμές των μεταβλητών που εκφράζουν την υγεία και την εμπειρία του παίκτη, καθώς και την ημερομηνία στην οποία διαδραματίζονται τα τρέχοντα γεγονότα της ιστορίας.

|  |
| --- |
| A computer screen shot of text  Description automatically generated  Εικόνα 39: Απόσπασμα της συνάρτησης DisplayAttributes(). |

7.5.2 Η συνάρτηση SpendXP()

Στην εικόνα [40] φαίνεται η συνάρτηση SpendXP(), η οποία καλείται όταν ο παίκτης πατάει τα αρμόδια κουμπιά στη σκηνή για να εξαργυρώσει τους πόντους που έχει συλλέξει και να αναβαθμίσει ορισμένες δεξιότητες. Αντίστοιχα με το προηγούμενο παράδειγμα, συλλέγονται οι τιμές των μεταβλητών που εκφράζουν τις δεξιότητες του παίκτη και εφόσον οι πόντοι εμπειρίας (XP) είναι αρκετοί, ο παίκτης μπορεί να αναβαθμίσει τις δεξιότητες που επιθυμεί. Σε περίπτωση που δεν επαρκούν οι πόντοι, εμφανίζεται ένα μήνυμα που τον ενημερώνει αναλόγως.

|  |
| --- |
| A computer screen shot of text  Description automatically generated  Εικόνα 40: Η συνάρτηση SpendXP(). |

|  |
| --- |
| Εικόνα 41: Στιγμιότυπο του παιχνιδιού με την καρτέλα ‘Attributes’. |

7.5.3 Η συνάρτηση DisplayStatements()

Παρόμοια φιλοσοφία ακολουθεί και η ανάκτηση μιας λίστας της Ink. Στην επόμενη εικόνα, παρατίθεται η συνάρτηση DisplayStatements(). Η προκείμενη συνάρτηση καλείται όταν ο παίκτης εμφανίζει το πανελ που παρουσιάζει καταθέσεις άλλων χαρακτήρων της ιστορίας και αποδεικτικά στοιχεία που έχει συλλέξει. Για ακόμη μια φορά καλείται η μέθοδος variablesState, ωστόσο εδώ αξίζει να αναφερθεί πως επιστρέφεται ένα αντικείμενο τύπου InkList που προέρχεται από την κλάση Dictionary, που σημαίνει πως το κάθε στοιχείο των λιστών statement\_List και evidence\_List περιέχει ένα κλειδί (key) και την τιμή της μεταβλητής (value). Στο πλαίσιο της εργασίας , η παραπάνω πληροφορία δεν χρησιμοποιήθηκε, ωστόσο κρίθηκε σημαντική η ειδοποιός διαφορά ως προς τον τύπο δεδομένων που επιστρέφει μια λίστα της Ink.

Πέραν αυτού, η συνάρτηση DisplayStatements() έπειτα εξετάζει τα παιδία ενός Game Object, τα οποία εμπεριέχουν τα στοιχεία των λιστών που ενδιαφέρουν τον παίκτη και αν στην παρούσα φάση είναι ενεργά στο αρχείο της Ink, τα εμφανίζει στον παίκτη.

|  |
| --- |
| A computer screen shot of a blue screen  Description automatically generated  Εικόνα 42: Η συνάρτηση DisplayStatements(). |

8. Αθέμιτες Πρακτικές σε Unity Projects

Συλλέγοντας πληροφορίες από διάφορες πηγές, μεταξύ των οποίων πολλές είναι ευρέως προσβάσιμες, όπως το Unity Forum και το StackOverFlow, διαπιστώθηκαν ορισμένα μοτίβα προβληματικών προγραμματιστικών πρακτικών που συναντώνται συχνά σε Unity Projects.

8.1 Προβλήματα Επίδοσης

8.1.1 Λανθασμένη διαχείριση της μεθόδου Update()

Η μέθοδος Update() πηγάζει από την κλάση MonoBehaviour και καλείται μια φορά για κάθε frame του gameplay. Χρησιμοποιείται κυρίως για εργασίες που πρέπει να ελέγχονται ή να ενημερώνονται συχνά, όπως για τις εισόδους του παίκτη (όπως πληκτρολόγηση, χρήση του ποντικιού ή χειριστηρίου), για κίνηση των αντικειμένων της σκηνής, animations και για λογική του παιχνιδιού, δηλαδή ελέγχους υποθετικής λογικής που πρέπει να πραγματοποιούνται διαρκώς (όπως ο έλεγχος της υγείας του χαρακτήρα). Επομένως, εφόσον οι μέθοδοι εντός της Update() καλούνται σε κάθε frame, αν αυτές περιέχουν υπολογιστικά περίπλοκες και δαπανηρές διεργασίες, η επίδοση του προγράμματος μπορεί να μειωθεί σημαντικά. Συνεπώς, κάτι τέτοιο συνίσταται να αποφεύγεται, όποτε είναι εφικτό.

Το φαινόμενο αλόγιστης χρήσης της Update() παρατηρείται συχνά μέσω δημιουργίας και καταστροφής αντικειμένων εντός αυτής. Οι δυο κύριες μέθοδοι που χρησιμοποιούνται για αυτή την εργασία είναι οι Instantiate() και Destroy(), oι οποίες όμως είναι δαπανηρές και η συμπερίληψή τους εντός της Update επιφέρει αρνητικά αποτελέσματα.

|  |
| --- |
| A screen shot of a computer code  Description automatically generated  Εικόνα 43: Παράδειγμα κακής χρήσης της Update(). (Πηγή: Antonio Borrelli, Gerardo Canfora, Vittoria Nardone, Guiseppe A. Di Lucca, Massimiliano Di Penta - Detecting Video Game-Specific Bad Smells in Unity Projects ) |

8.1.2 Η μέθοδος Find()

Μια ακόμη δυνητικά επιζήμιος πρακτική είναι η συχνή χρήση της μεθόδου Find(). Χρησιμοποιείται για την αναζήτηση ενός αντικειμένου της σκηνής, θέτοντας ως παράμετρο το όνομα που έχει δοθεί στο αντικείμενο εντός της σκηνής, όπως παρακάτω.

**GameObject obj = GameObject.Find("Player");**

Η προηγούμενη γραμμή κώδικα θα αναθέσει στο obj το αντικείμενο *Player*, αφού η μέθοδος αναζητήσει και βρει το αντικείμενο στη σκηνή.

Πρώτο μειονέκτημα είναι πως η Find() αναζητεί το αντικείμενο σε ολόκληρη τη σκηνή, επομένως σε μια σκηνή με πολλά αντικείμενα, η χρήση αυτής της μεθόδου θα κοστίσει στην απόδοση (πόσο μάλλον αν συμπεριληφθεί εντός της Update() ). Επιπλέον, το γεγονός ότι η λογική της εξαρτάται από ονόματα σε strings μπορεί εύκολα να προκαλέσει σύγχυση και ακαμψία σε περιπτώσεις που υπάρχουν αντικείμενα με το ίδιο όνομα ή αν αλλάξει το όνομα του αντικειμένου. Μια καλή εναλλακτική προσέγγιση σε περίπτωση που τα αντικείμενα προϋπάρχουν ήδη στη σκηνή και δε δημιουργούνται κατά το χρόνο εκτέλεσης είναι η αναφορά σε αυτά από το Inspector Window, μέσω συμπερίληψης του script που τα διαχειρίζεται ως component. Η παραπάνω εναλλακτική υιοθετήθηκε σε μεγάλο βαθμό στην παρούσα εργασία, όπου αυτό ήταν δυνατό.

|  |
| --- |
| A screenshot of a computer  Description automatically generated  Εικόνα 44: Αναφορά του αντικειμένου money στη σκηνή μέσω του Inspector. |

Σε περίπτωση που ένα αντικείμενο δημιουργείται κατά το χρόνο εκτέλεσης και το παραπάνω δεν είναι εφικτό, υπάρχει επίσης η μέθοδος FindWithTag(). Η μέθοδος αυτή αναζητεί το αντικείμενο βάσει ενός tag που έχει οριστεί για αυτό. Είναι προτιμότερη της Find(), διότι αναζητεί κατευθείαν τα αντικείμενα με το συγκεκριμένο tag και επιπλέον, η Unity αποθηκεύει και διαχειρίζεται τα tags αρκετά πιο αποδοτικά.

8.2 Προβλήματα Συντήρησης

Ένα script της MonoBehaviour που αναλαμβάνει ταυτόχρονα διαφορετικές ευθύνες κάνει τα projects δύσκολα στην περαιτέρω ανάπτυξη. Για παράδειγμα, έστω ότι ένα αντικείμενο εκφράζει έναν παίκτη (π.χ., ένα άτομο που περπατά σε έναν ανοιχτό κόσμο). Ένα τυπικό λάθος είναι να υλοποιηθεί όλη η λογική του παίκτη στο ίδιο scrip, δηλαδή να γίνεται η λήψη της εισόδου από το χειριστήριο, ο προσδιορισμός της κατάστασης του αντικειμένου (π.χ., αν μπορεί να περπατήσει, να πηδήξει, να πυροβολήσει, κτλ.), και η κίνηση του αντικειμένου.

Αντίθετα, συνιστάται να αποσπαστεί η διαχείριση των εισόδων σε ξεχωριστή ιεραρχία κλάσεων έτσι ώστε ο πηγαίος κώδικας που συνδέεται με το αντικείμενο του παίκτη να μην χρειάζεται να αλλάξει αν, για παράδειγμα, κάποιος θέλει να υλοποιήσει έναν παίκτη, τον οποίο χειρίζεται τεχνητή νοημοσύνη. Ομοίως, η διαχείριση της κατάστασης και η κίνηση του παίκτη πρέπει να χειρίζονται σε ξεχωριστά scripts.

Όπως έχει εξηγηθεί, στη Unity, είναι δυνατό να συζευχθεί ένα script της MonoBehaviour με άλλα αντικείμενα μέσω του IDE (Integrated Development Environment). Αυτό λειτουργεί ως εξής: Εάν ένα script δηλώσει public fields ή private/protected fields με το attribute [SerializedField], αυτά τα fields θα εμφανιστούν ως ιδιότητες του script στον Inspector. Στη συνέχεια, από το IDE, ο προγραμματιστής μπορεί απλώς να σύρει αντικείμενα σε αυτές τις ιδιότητες για να δημιουργήσει μια σύζευξη. Στο παράδειγμα της εικόνας [45], το script της MonoBehaviour έχει ορισμένα attributes [SerializedField]. Όπως φαίνεται στην εικόνα [46], αυτά εμφανίζονται ως ιδιότητες στον Inspector. Εφόσον ο προγραμματιστής έχει σύρει, για παράδειγμα, το Choice Button στο αντίστοιχο property, το πλαίσιο εμφανίζει ένα σύνδεσμο προς το αντικείμενο Choice Button.

Ένα αρνητικό παράγωγο υπό μια έννοια είναι η δυσκολία κατανόησης, δηλαδή οι συζεύξεις δεν θα είναι ορατές από τον πηγαίο κώδικα, αλλά μόνο από τον Inspector. Εάν ένας προγραμματιστής επεξεργαστεί τον πηγαίο κώδικα, ή μετονομάσει scripts ή αντικείμενα, οι συζεύξεις χάνονται και πρέπει να αποκατασταθούν χειροκίνητα. Υπάρχουν εναλλακτικές σε αυτό, συμπεριλαμβανομένης της χρήσης ενός συστήματος ανταλλαγής μηνυμάτων για τη σύζευξη διαφορετικών αντικειμένων στο παιχνίδι. Ωστόσο, τέτοιες εναλλακτικές μπορεί σε ορισμένες περιπτώσεις να είναι μη βέλτιστες όσον αφορά τις επιδόσεις. Στο πλαίσιο της εργασίας προτιμήθηκε αυτή η προσέγγιση, εφόσον βασίστηκε σε ατομική προσπάθεια.

|  |
| --- |
| A computer screen shot of a blue screen  Description automatically generated  Εικόνα 45: Χρήση του [SerializeField]. |

|  |
| --- |
| Εικόνα 46: Ανάθεση αντικειμένων στα πεδία που ορίζει το [SerializeField]. |

9. Το εργαλείο Midjourney

Τα τελευταία χρόνια, η επανάσταση της τεχνητής νοημοσύνης έχει επιτρέψει τη δημιουργία επαναστατικών και ταυτόχρονα προσβάσιμων εργαλείων και προγραμμάτων, τα οποία έχουν πλέον περάσει και στην οικιακή χρήση. Συγκεκριμένα, μια από τις πιο διαδεδομένες εκφάνσεις της οικιακής τεχνητής νοημοσύνης αφορά στον καλλιτεχνικό τομέα, και ιδίως στη δημιουργία εικόνων, μέσω ενός prompt. Ένα από τα πιο δημοφιλή εργαλεία αυτής της κατηγορίας είναι το Midjourney. Το συγκεκριμένο εργαλείο χρησιμοποιήθηκε σε μεγάλο βαθμό για το artwork του παιχνιδιού.

Η διαδικασία είναι πολύ απλή, καθώς με την είσοδο στον Discord server ή στην επίσημη ιστοσελίδα, ο χρήστης μπορεί να ξεκινήσει κατευθείαν να δημιουργεί prompts. Ένα απλό prompt, όπως το παρακάτω, μπορεί να δημιουργήσει τις επόμενες εικόνες :

**/imagine prompt: musician looking at his laptop excitedly**

|  |
| --- |
| A collage of a person playing guitar and singing  Description automatically generated  Εικόνα 47: Αποτελέσματα εικόνων του Midjourney (Πηγή: Curtis Pykes – DataCamp). |

Κάθε εκτέλεση ενός prompt του χρήστη επιστρέφει 4 εκδοχές της περιγραφής που ζητήθηκε. Ωστόσο, τα αποτελέσματα μπορούν να τροποποιηθούν. Ο χρήστης έχει στη διάθεση του ορισμένα κουμπιά που του επιτρέπουν τις εξής επεξεργασίες :

* Να κάνει upscaling σε μια εικόνα, δηλαδή αυτή να μεγεθυνθεί και να της προστεθούν επιπλέον λεπτομέρειες.
* Να δημιουργήσει παραλλαγές της εικόνας, την οποία επιλέγει από τις 4 αρχικές και να καθορίσει αν αυτές οι παραλλαγές θα είναι έντονες ή πιο διακριτικές.
* Να ζητήσει επανάληψη της διαδικασίας για διαφορετικά αποτελέσματα.

Είναι επίσης εφικτό να δοθούν επιπλέον παράμετροι στο prompt για πιο στοχευμένα αποτελέσματα. Οι παράμετροι μπορεί να αφορούν σε διαστάσεις της εικόνας, στην έκδοση του AI μοντέλου που χρησιμοποιείται και σε διάφορες άλλες επιλογές :

**/imagine prompt: musician looking at his laptop excitedly --ar 16:9 --v 4**

Μια από τις πιο ανεπτυγμένες δυνατότητες που προσφέρει είναι η προσθήκη άλλων εικόνων στο prompt ως αναφορές για ένα αντίστοιχο αποτέλεσμα, είτε μέσω URL, είτε από τη συλλογή του χρήστη. Επιπλέον, υπάρχει η δυνατότητα προσθήκης βάρους σε μια εικόνα για να καθοριστεί το πόσο θα επηρεαστεί το τελικό αποτέλεσμα από αυτή, γράφοντας στο τέλος --iw και δίπλα τον παράγοντα επιρροής (π.χ --iw 0.5). Το εργαλείο της Midjourney κρύβει πολλές ακόμα δυνατότητες, αλλά εδώ αναφέρθηκαν οι πιο διαδεδομένες. Ακολουθούν ορισμένες εικόνες του παιχνιδιού που δημιουργήθηκαν χάρη στο εργαλείο αυτό :

|  |
| --- |
| A person smoking a cigarette in front of a window  Description automatically generated  Εικόνα 48: Εικονογραφία του παιχνιδιού. |
| A police officer in a room  Description automatically generated  Εικόνα 49: Εικονογραφία του παιχνιδιού. |
| A person looking at a person in a dark room  Description automatically generated  Εικόνα 50: Εικονογραφία του παιχνιδιού. |

10. Το παιχνίδι *Beyond Man*

10.1 Η φιλοσοφία του παιχνιδιού

Η φιλοσοφία πίσω από την υλοποίηση του παιχνιδιού *Beyond Man* ήταν η δημιουργία ενός παιχνιδιού αλληλεπιδραστικού μυθιστορήματος, το οποίου το gameplay, παρόλο που βασίζεται σε πολύ μεγάλο βαθμό από την επιλογή μεταξύ ρητά διατυπωμένων ενεργειών ανά τακτά χρονικά διαστήματα, αντλεί ευρύτερα χαρακτηριστικά από την κατηγορία παιχνιδιών RPG (role-playing games). Μερικά αυτών αποτελούν την (σχετική) παροχή ελεύθερης περιήγησης στον παίκτη, επιτρέποντάς του να μεταφέρεται μεταξύ των χώρων ή μονοπατιών του παιχνιδιού και να τα εξερευνεί σύμφωνα με τη δική του κρίση και με τη σειρά που αυτός επιλέγει, χωρίς να του επιβάλλεται μια συγκεκριμένη γραμμική ροή. Σε αυτό συμβάλλει η προσπάθεια προσομοίωσης ενός world model.

Αρχικά, μέσω περιορισμών και κανόνων που επιβάλλονται στον παίκτη ανάλογα με την υπόστασή του και το εκάστοτε περιβάλλον του. Για παράδειγμα, έχει πεπερασμένο επίπεδο υγείας ή αν ο χαρακτήρας επιθεωρεί κρυφά ένα δωμάτιο, έχει περιορισμένο χρόνο μέχρι να επιστρέψει ο ιδιοκτήτης του και κάθε επιλογή του προσμετράται χρονικά. Κατά δεύτερον, με τη δυνατότητα διαμόρφωσης γνωρισμάτων του χαρακτήρα, μέσω ενός συστήματος που καθορίζεται μερικώς από το αρχείο της ιστορίας και εν μέρει από την εξωτερική παρέμβαση του χρήστη, μέσω άλλων scripts. Τα προσωπικά αυτά γνωρίσματα, ανάλογα με το επίπεδό τους μπορούν να παράξουν νέες και διαφορετικές επιλογές και εκβάσεις στην ιστορία. Τρίτον, είναι εφικτή η συλλογή αντικειμένων, τα οποία, εφόσον είναι στην κατοχή του παίκτη, αλληλεπιδρούν με το γύρω περιβάλλον και τους υπόλοιπους χαρακτήρες, δημιουργώντας ανάλογα αποτελέσματα.

Πέραν αυτής της προσέγγισης, υπήρξε η επιθυμία οπτικής και ακουστικής παρουσίασης του παιχνιδιού με έναν τρόπο που να κεντρίζει το ενδιαφέρον του παίκτη, χωρίς όμως να τον υποβάλλει σε περίπλοκες ενέργειες παιχνιδιού ή να είναι απαιτητικό από άποψη απόδοσης του υπολογιστή. Ουσιαστικά, να είναι ξεκούραστο, ευπαρουσίαστο και προσβάσιμο για κάθε επίπεδο παίκτη και υπολογιστικό σύστημα.

10.2 Σύνοψη του παιχνιδιού

Το παιχνίδι αναθέτει στον παίκτη το χειρισμό ενός αστυνομικού ντετέκτιβ, ο οποίος καλείται να εξιχνιάσει ένα μυστήριο. Προκειμένου να το καταφέρει αυτό, ο χαρακτήρας πρέπει να συλλέξει αντικείμενα και αποδεικτικά στοιχεία από τον τόπο του εγκλήματος, να συνομιλήσει με μάρτυρες και ύποπτους, έτσι ώστε να αποκτήσει πιο ξεκάθαρη εικόνα και γενικότερα, να ακολουθήσει ενδείξεις και μονοπάτια που θα τον οδηγήσουν εγγύτερα στη λύση της υπόθεσης.

Διαθέτει συγκεκριμένο αριθμό ημερών (παιχνιδιού), μέσα στον οποίο πρέπει να δράσει. Η διαχείριση αυτού του χρόνου είναι στην κρίση του παίκτη, καθώς αυτός θα επιλέξει σε ποιες ενέργειες θα δώσει προτεραιότητα, για παράδειγμα ποια πρόσωπα και τοποθεσίες θα επισκεφθεί και με ποια σειρά. Ύστερα, τα αντικείμενα που συλλέγει μπορούν να χρησιμοποιηθούν σε συγκεκριμένες στιγμές ή περιβάλλοντα και, ενδεχομένως, να φανερώσουν νέες πληροφορίες. Επίσης, ο παίκτης έχει στη διάθεσή του δυο κατηγορίες γνωρισμάτων. Τα γνωρίσματα προσωπικότητας (personality) και τα γνωρίσματα ικανοτήτων (skills). Η πρώτη κατηγορία τροποποιείται αυτόματα, ανάλογα με ορισμένες επιλογές του παίκτη, χωρίς όμως να ενημερώνεται εκείνη τη στιγμή ο παίκτης. Τα γνωρίσματα της δεύτερη κατηγορία μπορούν να τροποποιηθούν συνειδητά με την εξαργύρωση πόντων εμπειρίας (XP) στα πεδία που επιθυμεί ο παίκτης. Τα επίπεδα των περιεχομένων και των δυο κατηγοριών τροποποιούν συχνά το κείμενο της ιστορίας, τις επιλογές και τα μονοπάτια που εμφανίζονται. Για μια ακόμα πιο δυναμική και εμβυθιστική εμπειρία, πολλές επιλογές απαιτούν το άθροισμα του επιπέδου ενός γνωρίσματος και της τιμής ενός ζαριού να υπερβαίνει μια συγκεκριμένη τιμή για να χαρακτηριστεί επιτυχημένη. Ανάλογα με την έκβαση του αποτελέσματος, ακολουθούνται διαφορετικές εκβάσεις στην πλοκή. Όλα τα παραπάνω στοιχεία συντελούν στη διαφορετική εμπειρία του παιχνιδιού για κάθε παίκτη και να σε ενδεχόμενη επιθυμία επανάληψης για λήψη νέων αποφάσεων και στρατηγικών. Αξίζει να σημειωθεί ότι το παιχνίδι προσφέρει πολλαπλά τέλη της ιστορίας.

10.3 Έλεγχος Περατότητας

Όπως ενδεχομένως να έχει γίνει αντιληπτό, μια ιστορία πολλαπλών επιλογών δημιουργεί πολλές διαδοχικές διακλαδώσεις, οι οποίες σε συνδυασμό με χρήση επιπλέον λογικών συνθηκών καθιστούν αρκετά απαιτητική έως αδύνατη την παρακολούθηση όλων των πιθανών μονοπατιών και συνεπώς, τον έλεγχο περατότητας. Ο έλεγχος αυτός αφορά σε περιπτώσεις όπου η ιστορία κατά λάθος εγκλωβίζεται σε ατέρμονες βρόχους (infinite loops) ή οδηγείται σε αδιέξοδο, δηλαδή σε σημεία όπου δεν προοριζόταν να ολοκληρωθεί, είτε λόγω συντακτικών, είτε λογικών σφαλμάτων. Για τον έλεγχο τέτοιων περιπτώσεων υλοποιήθηκε το *RandomInkTraverser*, ένα script που σε κάθε εκτέλεση καλείται να διασχίσει την ιστορία, πραγματοποιώντας μια τυχαία επιλογή σε κάθε κόμβο που συναντάει. Αν συναντήσει κάτι που αντιλαμβάνεται ως βρόχο (το όριο των επαναλήψεων μπορεί να καθοριστεί), είτε απρόσμενο σφάλμα, επιστρέφει και ειδοποιεί. Διαφορετικά, ενημερώνει ότι η ιστορία τερματίστηκε κανονικά. Παράλληλα, τυπώνει στο τερματικό κάθε τμήμα κειμένου που συναντά, έτσι παρέχει μια ξεκάθαρη εικόνα του συνολικού μονοπατιού που ακολουθήθηκε και ο εντοπισμός τυχόν σφάλματος είναι εύκολος. Είναι σημαντικό να επαναληφθεί μεγάλο αριθμό φορών αυτή η διαδικασία, πιθανόν και με τροποποιήσεις των μεταβλητών του παιχνιδιού για ελαχιστοποίηση του ενδεχομένου απρόβλεπτων συμπεριφορών.

11. Επίλογος

Σκοπός της εργασίας ήταν, σε πρώτο στάδιο, η επαφή με τα διάφορα authoring systems που χρησιμοποιούνται για τη δημιουργία παιχνιδιών interactive fiction, και κατόπιν, η ανάπτυξη ενός παιχνιδιού, χρησιμοποιώντας ένα από αυτά. Για το λόγο αυτό, η αρχική φάση της εργασίας περιλάμβανε τη μελέτη και δοκιμή διάφορων γνωστών systems, όπως τα Inform 7, TADS 3, Twine, Hugo, Inky κλπ., ως προς τα εργαλεία που παρέχουν στον προγραμματιστή και τις γλώσσες προγραμματισμού που τα συνοδεύουν, ενώ αφιερώθηκε χρόνος και στη δοκιμή κυκλοφοριών έργων interactive fiction που αναπτύχθηκαν με τα παραπάνω συστήματα.

Η επιλογή της Ink έγινε με βάση την πεποίθηση ότι η choice-based δομή που παρέχει, θα επιτρέψει την ανάπτυξη ενός παιχνιδιού που ενδεχομένως να έχει μεγαλύτερη απήχηση στον μέσο παίκτη, σε αντίθεση με τα παιχνίδια που χρησιμοποιούν text parser και καλούν τον παίκτη να πληκτρολογεί διαρκώς. Έπειτα, η σύνταξή της, καθώς και ο Inky Editor, παρέχουν μια ξεκούραστη και ευχάριστη εμπειρία στον προγραμματιστή και τον απαλλάσσουν από την ανάγκη να συντάσσει κλάσεις και δομές, προσφέροντάς του περισσότερο χρόνο να προγραμματίσει τη λογική της ιστορίας. Ύστερα, τα διαθέσιμα πακέτα για ενσωμάτωση .ink αρχείων σε projects μεγάλων game engines, όπως Unity και Unreal, προσέφεραν τη δυνατότητα εξοικείωσης με αυτά, και με ευρέως διαδεδομένες πρακτικές του αντικειμενοστραφούς προγραμματισμού. Τέλος, η χρήση του Unity game engine παρείχε απεριόριστες δυνατότητες ως προς την παρουσίαση της ιστορίας.

Προσωπικός στόχος ήταν η δημιουργία ενός εύχρηστου και παραστατικού παιχνιδιού με ενδιαφέρουσα πλοκή σύνθετων διακλαδώσεων που, παρά το πολύ απλό gameplay, δίνει στον παίκτη σημαντική εξουσία και ελευθερία. Επιπλέον, υπήρχε η επιθυμία για ένα παιχνίδι προσβάσιμο για κάθε χρήστη που χειρίζεται σε ικανοποιητικό επίπεδο την αγγλική γλώσσα και έχει στη διάθεσή του έστω και έναν απλό επιτραπέζιο υπολογιστή ή λαπτοπ, μη υψηλών απαιτήσεων. Η παρούσα αναφορά αποσκοπούσε, πρώτον, στην εισαγωγή του μέσου αναγνώστη στη φιλοσοφία του interactive fiction και στην ανάδειξη τρόπων που μπορεί και ο ίδιος να γράψει μια ιστορία και, δεύτερον, σε μια πιο τεχνική επεξήγηση του τρόπου με τον οποίο υλοποιήθηκε το παιχνίδι *Beyond Man*.

**ΠAΡΑΡΤΗΜΑ**

* To GitHub Repository με τον κώδικα που αναπτύχθηκε για το παιχνίδι:

<https://github.com/cdervs/BeyondMan>

* To παιχνίδι μπορεί να βρεθεί στον παρακάτω σύνδεσμο:

<https://www.dropbox.com/scl/fo/b35apne0zin30qof2zudx/ALDf_QsdebR1fdRVOQbjTsI?rlkey=bd7fdz0g971iyj2tn3w1u8lgz&st=1y04qga3&dl=0>

**ΒΙΒΛΙΟΓΡΑΦΙΑ**

1. Techopedia - *What does parser mean ?* <https://www.techopedia.com/definition/3854/parser>
2. Ben Lutkevick (TechTarget) - *Parser* <https://www.techtarget.com/searchapparchitecture/definition/parser>
3. Jonathan Kotchian - *Teaching Composition with Interactive Fiction*<https://techstyle.lmc.gatech.edu/teaching-composition-with-interactive-fiction/>
4. IFWiki - *Building a new Interactive Fiction System* <https://www.ifwiki.org/Building_a_New_Interactive_Fiction_System>
5. IFWiki - *Choice-based interactive fiction*

<https://www.ifwiki.org/Choice-based_interactive_fiction>

1. Anna Washenko - *From Infocom to 80 Days: An oral history of text games and interactive fiction*

<https://arstechnica.com/gaming/2024/06/from-infocom-to-80-days-an-oral-history-of-text-games-and-interactive-fiction/>

1. The Interactive Fiction Database

<https://ifdb.org/viewlist?id=tyypotkxpx3g2wl5>

1. Inform 6 Official Page

<https://www.inform-fiction.org/>

1. Z-machine – Wikipedia

<https://en.wikipedia.org/wiki/Z-machine>

1. TADS 3 Official Page

<https://tads.dev/index.html>

1. Twine Cookbook - Story formats

<https://twinery.org/cookbook/introduction/story_formats.html#:~:text=Story%20Format%3A%20Dialects,-In%20Twine%201&text=Now%2C%20story%20formats%20are%20more,to%20write%20they%20own%20functionality>.

1. The Interactive Fiction Technology Foundation - *Our Mission and Goals*

<https://iftechfoundation.org/mission/>

1. InkleStudios - Ink: A narrative scripting language for games

<https://www.inklestudios.com/ink/>

1. JK Lucent - *Top 15 Reasons to Use Unity 3D for Game Development*

<https://www.linkedin.com/pulse/top-15-reasons-use-unity-3d-game-development-jklucent-cvhjc>

1. Unity Learn - *Explore the Unity Editor*

<https://learn.unity.com/tutorial/explore-the-unity-editor-1#>

1. Medium - *Object-Oriented Programming (OOP) – Unity C#*

<https://imran-momin.medium.com/object-oriented-programming-oop-unity-c-aa0179999e85>

1. Lenovo - *What is an instance ?*

<https://www.lenovo.com/us/en/glossary/what-is-instance/?orgRef=https%253A%252F%252Fwww.google.com%252F>

1. Unity Documentation

<https://docs.unity3d.com/Manual/index.html>

1. Midjourney – Wikipedia

<https://en.wikipedia.org/wiki/Midjourney>

1. Nick Bucher (The University of Alabama) - *Introducing Design Patterns and Best Practices in Unity*
2. Chris Proctor, Paulo Blinkstein (Stanford University) - *Interactive Fiction: Weaving together literacies of text and code*
3. James Dean Palmer (Northern Arizona University) - *Ficticious: MicroLanguages for Interactive Fiction*
4. David E. Millard (University of Southampton) - *Games/Hypertext*
5. Jérémy Frey, Gilad Ostrin, May Grabli, Jessica R. Cauchard - *Physiologically Driver Storytelling: Concept and Software Tool*
6. Jacob Garbe, Max Kreminski, Ben Samuel - *StoryAssembler: An Engine for Generating Dynamic Choice-Driven Narratives*
7. Antonio Borrelli, Gerardo Canfora, Vittoria Nardone, Guiseppe A. Di Lucca, Massimiliano Di Penta - *Detecting Video Game-Specific Bad Smells in Unity Projects*