SAE CREATIVE MEDIA COLLEGE

Πρόταση Πτυχιακής Εργασίας

Πρόγραμμα Σπουδών: BA/BSc in Games Programming
Μάθημα: Research Practice and Society - CMN6100 23S2 [ATH]
Κωδικός Εργασίας: CMN6100.S3
Όνομα & Κωδικός σπουδαστή: Μιχαήλ-Ευάγγελος Διαμαντής, 13-13246
Ημ/νία Υποβολής: 25/2/24

Περιεχόμενα

Τίτλος

The “Snapshot”: A complete, full-fledged and open source video game C++ data serialization solution

Εισαγωγή

Στην επικείμενη πτυχιακή εργασία, θα γίνει μια προσπάθεια σχεδιασμού και υλοποίησης ενός C++ library, με το FFI(wiki.haskell.org, n.d.) του και για την προβολή των αποτελεσμάτων θα γίνει χρήση της τελικής βιβλιοθήκης σε μια σκηνή με απλά δεδομένα στη μηχανή παιχνιδιών Unity(Unity.com, 2023).
Ο απώτερος σκοπός είναι να χτιστεί αυτή η βιβλιοθήκη με γνώμονα να γίνει open source(Laurent, A.M.S, 2004) και να δημιουργήσει την υποδομή για ένα πλήρες saving system solution(The Strong National Museum of Play, 2011) το οποίο ο εκάστοτε προγραμματιστής θα μπορεί εύκολα και γρήγορα να εντάξει στην εκάστοτε μηχανή παιχνιδιών για χρήση στο δικό του project.
Πιο συγκεκριμένα, θα δημιουργηθεί μια εκτενής βιβλιοθήκη σε μορφή dll συνοδευόμενο από ένα Foreign Function Interface που θα του δώσει τη δυνατότητα χρήσης αυτής της βιβλιοθήκης σε εξωτερικά προγράμματα και μηχανές παιχνιδιών. Με την εσωτερική χρήση serializer(Ltd, C. and Hermans K., 2023) τα δεδομένα και οι δομές που θα έχει δημιουργήσει ο προγραμματιστής θα μπορέσουν να αποθηκευτούν στο σημείο επιλογής του σε binary (Ltd, C. and Hermans K., 2023) μορφή προσθέτοντας τους έτσι και μιας τυπικής μορφής προστασία, ωστόσο αυτό βγαίνει εκτός των ορίων της πτυχιακής αυτής.
Για την περάτωση όλων των παραπάνω θα γίνει μια ενδελεχής μελέτη ολοκληρωμένων συστημάτων, εάν υπάρχουν, ώστε να καταγράφουν οι τρόποι σχεδιασμού τους και οι τεχνικές με τους οποίους διαχειρίζονται τα αποθηκευμένα δεδομένα. Θα αναλυθούν διαφορετικών τύπων serializers ώστε να βρεθεί ο πιο optimal όσο αφορά το performance και τέλος θα γίνει μια ενδελεχής ανάλυση ως προς τις δομές δεδομένων που θα χρησιμοποιεί εσωτερικά η βιβλιοθήκη αυτή για τη σωστή και αποτελεσματική διαχείριση των αποθηκευμένων δεδομένων. Επιπλέον, θα δημιουργηθεί ένα σύστημα διατήρησης των references μεταξύ των κλάσεων των αποθηκευμένων τύπων μέσα στο εσωτερική backing δομή της βιβλιοθήκης.

Σκεπτικό και σημασία

Η ιστορία της αποθήκευσης και της τροποποίησης δεδομένων σε βιντεοπαιχνίδια είναι ένα συναρπαστικό ταξίδι που ακολουθεί την εξέλιξη της τεχνολογίας των παιχνιδιών και των προσδοκιών των παικτών. Στις πρώτες ημέρες των βιντεοπαιχνιδιών, τα συστήματα αποθήκευσης ήταν υποτυπώδη, συχνά περιορισμένα σε κωδικούς πρόσβασης ή κωδικούς που μπορούσαν να εισάγουν οι παίκτες για να συνεχίσουν την πρόοδό τους(AtariAge Forums, 2006).

Αυτά τα συστήματα χρησιμοποιήθηκαν κυρίως για να παρακάμψουν τους περιορισμούς του υλικού και να προσφέρουν στους παίκτες έναν τρόπο να συνεχίσουν τις περιπέτειές τους σε πολλαπλά playthrough. Καθώς η τεχνολογία των παιχνιδιών εξελισσόταν, το ίδιο συνέβαινε και με τα συστήματα αποθήκευσης, με την εισαγωγή κασετών μνήμης με μπαταρία και εσωτερικής μνήμης (Micheloni, Crippa and Marelli, 2010). Αυτό επέτρεψε στους προγραμματιστές να εφαρμόσουν πιο εξελιγμένα συστήματα αποθήκευσης, όπως σημεία αποθήκευσης και χειροκίνητες αποθηκεύσεις, παρέχοντας στους παίκτες μεγαλύτερο έλεγχο της προόδου τους.
Με την εξέλιξη της τεχνολογίας και την δημιουργία αντικειμενοστραφής γλωσσών προγραμματισμού έκανε την εμφάνιση του ένα σχετικά πιο σύνθετο πρόβλημα στον τομέα των βιντεοπαιχνιδιών, αυτό της αποθήκευσης των σχέσεων μνήμης μεταξύ των δημιουργούμενων instances των κλάσεων που έχει γράψει ο εκάστοτε προγραμματιστής. Κατά καιρούς, έχουν υπάρξει διάφορες υλοποιήσεις και τεχνικές για την καταγραφή των σχέσεων αυτών είτε σε απλά JSON/XML(Ltd, C. and Hermans K., 2023) αρχεία, σε βάσεις δεδομένων αλλά και σε binary μορφές. Φυσικά στη διαδικασία αυτή το performance των χρησιμοποιουμένων δομών δεδομένων αλλά και του serializer της εκάστοτε τεχνικής παίζει μεγάλο ρόλο στους χρόνους καταγραφής των δεδομένων.
Στόχος της πτυχιακής αυτής εργασίας είναι η συγγραφή ενός open source native language library σε C++ συνοδευόμενο από ένα FFI (Foreign Function Interface) στο οποίο ο εκάστοτε προγραμματιστής θα μπορεί να δημιουργήσει έναν custom wrapper (Gorpynich, O. (2023) για τη δική του χρήση ώστε να εντάξει εύκολα και γρήγορα ένα ολοκληρωμένο saving σύστημα μέσα στο παιχνίδι του.
Μέσα από την εξέταση παλαιοτέρων τρόπων αποθήκευσης δεδομένων, τη δημιουργία και χρήση ενός ID custom συστήματος για την διατήρηση των σχέσεων μεταξύ των κλάσεων κατά το serialization αλλά και το deserialization, την εξέταση της πιο optimal μορφής αποθήκευσης δεδομένων και την χρήση κατάλληλων δομών δεδομένων θα χτιστεί μια ολοκληρωμένη βιβλιοθήκη ικανή να συνδεθεί με αρκετές άλλες γλώσσες όπως Python, C# και Java.
Μέσα από την παραπάνω διατριβή θα αποκτηθεί ένας τεράστιος όγκος γνώσης επάνω στον τομέα της διαχείρισης μνήμης, την διαχείριση δεδομένων, το data serialization/deserialization αλλά και το τελικό αποτέλεσμα θα είναι η χρήση της δημιουργημένης αυτής βιβλιοθήκης σε μια μικρή σκηνή μέσα στη μηχανή παιχνιδιών Unity μέσω της δημιουργίας ενός C# wrapper για την μεταξύ τους διεπαφή.

Καθορισμός πλαισίου

Στον τομέα της ανάπτυξης βιντεοπαιχνιδιών, η διαχείριση και ο χειρισμός των δεδομένων παραμένει ένα αρκετά σύνθετο πρόβλημα. Από την αποθήκευση της προόδου των παικτών έως τη διαχείριση των assets του παιχνιδιού, το αποτελεσματικό serialization των δεδομένων παραμένει ένα σημαντικό κομμάτι για την εμπειρία του χρήστη αλλά και για την ομάδα ανάπτυξης του. Ωστόσο, παρά την πληθώρα μορφών data serialization (Ltd, C. and Hermans K.,

2023), όπως JSON, XML και binary, η ενσωμάτωση μιας ολοκληρωμένης λύσης στις μηχανές παιχνιδιών παραμένει μια αξιοσημείωτη πρόκληση λόγο της πολυπλοκότητας της.

Το data serialization(Ltd, C. and Hermans K., 2023) είναι η διαδικασία μετατροπής πολύπλοκων δομών δεδομένων ή καταστάσεων αντικειμένων σε μια μορφή που μπορεί εύκολα να αποθηκευτεί (serialization) ή να μεταδοθεί και να ανακατασκευαστεί (deserialization) αργότερα. Κάθε μορφή serialization έχει τα πλεονεκτήματά της και τις περιπτώσεις χρήσης της. Για παράδειγμα, το JSON (JavaScript Object Notation) είναι αναγνώσιμο από τον άνθρωπο και υποστηρίζεται ευρέως, γεγονός που το καθιστά δημοφιλές. Επίσης, η human-readable μορφή του το καθιστά και αρκετά εύκολο στο debugging . Η XML (eXtensible Markup Language) προσφέρει ιεραρχική οργάνωση δεδομένων και χρησιμοποιείται συνήθως για αρχεία ρυθμίσεων και ανταλλαγή δεδομένων μεταξύ διαφορετικών συστημάτων. Το binary serialization, από την άλλη πλευρά, είναι πιο συμπαγής και αποδοτικό όσον αφορά την αποθήκευση και τη μετάδοση αρχείων, καθιστώντας την κατάλληλη για εφαρμογές με κρίσιμες επιδόσεις, όπως τα παιχνίδια.

Παρά την ευελιξία αυτών των μορφών αυτών, η απρόσκοπτη ενσωμάτωσή τους στις μηχανές παιχνιδιών παρουσιάζει αρκετές προκλήσεις:

1. Απόδοση: Οι μηχανές παιχνιδιών συχνά απαιτούν επεξεργασία δεδομένων υψηλής απόδοσης για τη διατήρηση ομαλών framerate. Ενώ μορφές όπως το JSON και η XML προσφέρουν αναγνωσιμότητα και δια-λειτουργικότητα, η βασισμένη σε κείμενο φύση τους επιβαρύνει την απόδοση τους κατά τη διάρκεια της ανάλυσης και του serialization. Το binary format είναι πιο αποδοτικό, αλλά δεν είναι σε human-readable μορφή.

2. Platform Independence: Η ανάπτυξη παιχνιδιών εκτείνεται σε πολλαπλές πλατφόρμες, συμπεριλαμβανομένων PC, κονσολών και κινητών συσκευών. Μια ισχυρή λύση data serialization και saving για μηχανές παιχνιδιών θα πρέπει να είναι ανεξάρτητη από πλατφόρμες για να διασφαλίζεται η συμβατότητα σε διαφορετικά περιβάλλοντα (www.sdd-technology.com. n.d.).

3. Integration Complexity: Η ενσωμάτωση βιβλιοθηκών serialization τρίτων σε μηχανές παιχνιδιών μπορεί να είναι πολύπλοκη και χρονοβόρα. Οι προγραμματιστές παιχνιδιών μπορεί να χρειαστεί να γράψουν είτε δικές τους ολοκληρωμένες λύσεις μαζί με το περιβάλλον επικοινωνίας ή wrappers για να γεφυρώσουν το χάσμα μεταξύ της βιβλιοθήκης του serialization και των δομών δεδομένων της μηχανής.

4. Reference Preservation: Κατά τη διάρκεια του serialization είναι λογικό πως οι σχέσεις μεταξύ των κλάσεων θα χαθούν διότι δεν υπάρχει κάποιος τρόπος να αποθηκευτεί αυτούσια η ίδια η διεύθυνση της μνήμης εκείνης της στιγμής. Ένα σύστημα που κάνει χρήση ενός

εσωτερικού συστήματος IDing θα μπορεί εύκολα να ξαναχτίσει τις σχέσεις αυτές κατά τη διάρκεια του deserialization.

Με γνώμονα το δεδομένο πως δεν υπάρχει μέχρι στιγμής μια τέτοια ολοκληρωμένη λύση αλλά και την πληθώρα διαφορετικών data serializers που συνοδεύονται από διαφορετικούς wrapper για να μπορέσουν να ενταχθούν σε οποιοδήποτε πρόγραμμα, το σύστημα το οποίο θα δημιουργηθεί στα πλαίσια αυτής της εργασίας θα προσπαθήσει να γεφυρώσει αυτό το κενό με μια γρήγορη βιβλιοθήκη γραμμένη σε C++.
Η επιλογή της C++ ως η βασική γλώσσα της βιβλιοθήκης, όπου αναφέρεται συχνά ως "native language", έγινε επειδή επιτρέπει τον χειρισμό πόρων hardware σε low-level (Goodrich, M.T., Tamassia, R. and Mount, 2011) και την άμεση αλληλεπίδραση με την υποκείμενη αρχιτεκτονική του συστήματος. Αυτό σημαίνει ότι ο κώδικας της C++ μπορεί να μεταγλωττιστεί σε κώδικα μηχανής που είναι συγκεκριμένος για την αρχιτεκτονική του επεξεργαστή του συστήματος στο οποίο εκτελείται, επιτρέποντας την αποτελεσματική εκτέλεση χωρίς την ανάγκη για ξεχωριστό runtime environment ή virtual environment/machine.

Αντίθετα, γλώσσες όπως η Java, C# ή η Python αναφέρονται συχνά ως "διερμηνευμένες"(interpeted) ή "διαχειριζόμενες"(managed) γλώσσες (kb.iu.edu, n.d.), επειδή συνήθως εκτελούνται πάνω από ένα virtual environment ή έναν interpreter, ο οποίος αφαιρεί τις λεπτομέρειες της επαφής με το hardware και του λειτουργικού συστήματος για λόγους ευκολίας. Αν και αυτό μπορεί να προσφέρει πλεονεκτήματα όπως η ανεξαρτησία πλατφόρμας και η αυτόματη διαχείριση μνήμης, μπορεί επίσης να εισάγει κάποια επιβάρυνση απόδοσης σε σύγκριση με τις native languages όπως η C++.

Μεθοδολογία

Για την επιτυχής διεκπεραίωση αυτής της πτυχιακής εργασίας θα γίνει αρχικά μια ενδελεχής εξέταση παλαιών τρόπων και συστημάτων αποθήκευσης του progress του παίκτη σε ένα παιχνίδι. Στη συνέχεια θα αντληθούν από αυτούς πιθανοί τρόποι σχεδιασμού της βιβλιοθήκης που θα συνταχθεί σε αυτή την πτυχιακή. Κατά την φάση της εξέτασης των εργαλείων θα αναλυθούν οι ήδη υπάρχοντες τρόποι data serialization και deserialization και θα διατυπωθεί και σχεδιαστεί ο μηχανισμός διατήρησης των references κατά τη διάρκεια του αυτών. Επιπλέον, θα εξεταστεί η version της C++ που θα χρησιμοποιηθεί, οι δομές δεδομένων που θα πλαισιώσουν το back-end της εφαρμογής και τέλος θα εξεταστεί και επιλεχθεί ο binary formatter που θα κάνει τη μετατροπή των δεδομένων σε binary μορφή.
Στο τεχνικό στάδιο της πτυχιακής εργασίας θα γίνει, με τη χρήση τεχνικών σχεδιασμού back-end, ο πλήρης σχεδιασμός της C++ βιβλιοθήκης, ο πλήρης σχεδιασμός του FFI της βιβλιοθήκης για να μπορέσει το παραχθείς dll να χρησιμοποιηθεί από άλλες γλώσσες με κύριο στόχο τη C#

και τέλος ο πλήρης σχεδιασμός του C# wrapper για τη χρήση της βιβλιοθήκης στη μηχανή παιχνιδιών Unity.
Τέλος, θα υλοποιηθούν οι παραπάνω σχεδιασμοί με τα κατάλληλα εργαλεία και θα γίνει χρήση του dll σε μια απλή, με απλά δεδομένα, σκηνή στη μηχανή Unity και θα εξεταστούν τα αποτελέσματα.

Πόροι

Για την επιτυχής ολοκλήρωση αυτής της διατριβής θα εξεταστούν και χρησιμοποιηθούν τα παρακάτω εργαλεία:

- Visual Studio 2022
- Visual Studio Code
- C# 8.0 LTS
- C++ (η version θα αναλυθεί για compatibility)
- Unity 2023.x.x

Εργαλεία και προγράμματα προς εξέταση:

- Cap n’ Proto C/C++
- Protobuffs C/C++
- MessagePack for C/C++
- SWIG C/C++/C#

Παράλληλα με τα παραπάνω εργαλεία έχουν συλλεχθεί και θα συλλεχθούν επιστημονικά άρθρα, βιβλία και επιπλέον υποστηρικτικό υλικό για την χρήση τους στην εκάστοτε ανάλυση.

Πίνακας περιεχομένων

   a. Εξέταση ήδη υλοποιημένων τρόπων data serialization και deserialization
   b. Διατύπωση και σχεδιασμός για την επίλυση σχετικά με το reference preservation κατά το serialization και deserialization
   c. Ανάλυση ως προς την χρήση της C++
   d. Ανάλυσή ως προς την επιλογή του binary formatter που θα χρησιμοποιηθεί
   e. Ανάλυση και εξέταση των δομών δεδομένων που θα οδηγήσουν σε ένα optimal performance
3. Σχεδιασμός βιβλιοθήκης
   a. Σχεδιασμός της δομής του C++ library
   b. Σχεδιασμός του FFI του library για υποστήριξη από άλλες γλώσσες
   c. Σχεδιασμός του C# wrapper για χρήση της βιβλιοθήκης στη μηχανή Unity
4. Τεχνικό κομμάτι
   a. Υλοποίηση του κύριου C++ library
   b. Υλοποίηση του FFI
   c. Υλοποίηση του C# wrapper
5. Αποτελέσματα και συμπεράσματα
   a. Χρήση του library στη Unity
6. Βιβλιογραφία

Χρονοδιάγραμμα εκπόνησης πτυχιακής εργασίας

Διαμαντής Μιχαήλ-Ευάγγελος	Μάρτιος (Εβδ.)				Απρίλιος (Εβδ.)				Μάιος (Εβδ.)				Ιούνιος (Εβδ.)				Ιούλιος (Εβδ.)			
13-13246	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
Συλλογη πληροφοριων																				
Μελετη βιβλιογραφιας																				
Εξέταση και ανάλυση εργαλείων																				
Σχεδιασμός βιβλιοθήκης																				
Υλοποίηση του σχεδιασμού																				
Εξέταση αποτελεσμάτων																				
Συγγραφή πτυχιακής εργασίας																				

Βιβλιογραφία

Goodrich, M.T., Tamassia, R. and Mount, D.M. (2011). Data Structures and Algorithms in C++. 2nd edition ed. [online] Amazon. Hoboken, NJ: Wiley. Available at: https://www.amazon.com/Data-Structures-Algorithms-Michael-Goodrich/dp/0470383275.

Ltd, C. and Hermans, K. (2023). Mastering Data Serialization and Formats: A Comprehensive Guide to Learn Data Serialization and Formats. [online] Amazon. Independently published. Available at: https://www.amazon.com/Mastering-Data-Serialization-Formats-Comprehensive/dp/B0CK3ZX155/ref=tmm_pap_swatch_0?_encoding=UTF8&qid=&sr= [Accessed 21 Feb. 2024].

Lippman, S., Lajoie, J. and Moo, B. (2012). C++ Primer. 5th edition ed. [online] Amazon. Upper Saddle River, NJ Munich: Addison-Wesley Professional. Available at: https://www.amazon.com/Primer-5th-Stanley-B-Lippman/dp/0321714113 [Accessed 21 Feb. 2024].

www.sciencedirect.com. (n.d.). Digital Design and Computer Architecture | ScienceDirect. [online] Available at: https://www.sciencedirect.com/book/9780128000564/digital-design-and-computer-architecture.

Richards, M. and Ford, N. (2020). Fundamentals of Software Architecture: An Engineering Approach. 1st edition ed. [online] Amazon. Beijing Boston Farnham Sebastopol Tokyo: O'Reilly Media. Available at: https://www.amazon.com/Fundamentals-Software-Architecture-Comprehensive-Characteristics/dp/1492043451 [Accessed 21 Feb. 2024].

www.sciencedirect.com. (n.d.). ARM System Developer's Guide | ScienceDirect. [online] Available at: https://www.sciencedirect.com/book/9781558608740/arm-system-developers-guide.

ScienceDirect. (n.d.). ARM 64-Bit Assembly Language. [online] Available at: https://www.sciencedirect.com/book/9780128192214/arm-64-bit-assembly-language.

Boyd, C. (2020). A brief history of video game saves and data modification | Malwarebytes Labs. [online] Malwarebytes. Available at: https://www.malwarebytes.com/blog/news/2020/06/a-brief-history-of-video-game-saves-and-data-modification [Accessed 21 Feb. 2024].

Yallop, J., Sheets, D. and Madhavapeddy, A. (2018). A modular foreign function interface. Science of Computer Programming, 164, pp.82–97. Doi:https://doi.org/10.1016/j.scico.2017.04.002.

crystalnet-tech.com. (n.d.). Binary Serialization. [online] Available at: https://crystalnet-tech.com/RuntimeLibrary/Help/html/scr/Binary_Serialization.html [Accessed 21 Feb. 2024].

capnproto.org. (n.d.). Cap'n Proto: Introduction. [online] Available at: https://capnproto.org/ [Accessed 21 Feb. 2024].

JGup (2023). Choosing the save system for your game - Microsoft Game Development Kit. [online] learn.microsoft.com. Available at: https://learn.microsoft.com/en-us/gaming/gdk/_content/gc/system/overviews/game-save/game-saves-choosing-save-system [Accessed 21 Feb. 2024].

Wikipedia. (2024). Comparison of data-serialization formats. [online] Available at: https://en.wikipedia.org/wiki/Comparison_of_data-serialization_formats [Accessed 21 Feb. 2024].

arvindpdmn, anuradhac (2019). Data Serialization. [online] Devopedia. Available at: https://devopedia.org/data-serialization

URI (2019). How The Computer Works: The CPU and Memory. [online] Uri.edu. Available at: https://homepage.cs.uri.edu/faculty/wolfe/book/Readings/Reading04.htm

Kelley, M. (2021). How to use a C++ Library from C#. [online] XRLO—eXtended Reality Lowdown. Available at: https://medium.com/xrlo-extended-reality-lowdown/how-to-use-a-c-library-from-c-1549c3c48c0a [Accessed 21 Feb. 2024].

TylerMSFT (2021). How to: Wrap Native Class for Use by C#. [online] learn.microsoft.com. Available at: https://learn.microsoft.com/en-us/cpp/dotnet/how-to-wrap-native-class-for-use-by-csharp?view=msvc-170 [Accessed 21 Feb. 2024].

www.microsoftpressstore.com. (n.d.). Implement data access | Microsoft Press Store. [online] Available at: https://www.microsoftpressstore.com/articles/article.aspx?p=2931576&seqNum=4 [Accessed 21 Feb. 2024].

msgpack.org. (n.d.). MessagePack: It’s like JSON. but fast and small. [online] Available at: https://msgpack.org/index.html

GitHub. (2024). mono/CppSharp. [online] Available at: https://github.com/mono/CppSharp [Accessed 21 Feb. 2024].

Unity Blog. (n.d.). Persistent data: How to save your game states and settings. [online] Available at: https://blog.unity.com/games/persistent-data-how-to-save-your-game-states-and-settings

www.gamedeveloper.com. (n.d.). Saving the Day: Save Systems in Games. [online] Available at: https://www.gamedeveloper.com/design/saving-the-day-save-systems-in-games [Accessed 21 Feb. 2024].

Protocol Buffers Documentation. (n.d.). Protocol Buffers. [online] Available at: https://protobuf.dev/

www.swig.org. (n.d.). Simplified Wrapper and Interface Generator. [online] Available at: https://www.swig.org/

Gorpynich, O. (2023). What is FFI (Foreign Function Interface)? An Intuitive Explanation. [online] Medium. Available at: https://levelup.gitconnected.com/what-is-ffi-foreign-function-interface-an-intuitive-explanation-7327444e347a [Accessed 21 Feb. 2024].

Microsoft (2019). Visual Studio. [online] Visual Studio. Available at: https://visualstudio.microsoft.com/

Unity Technologies (2019). Unity. [online] Unity. Available at: https://unity.com/

wiki.haskell.org. (n.d.). FFI Introduction - HaskellWiki. [online] Available at: https://wiki.haskell.org/FFI_Introduction [Accessed 23 Feb. 2024].

Laurent, A.M.S. (2004). Understanding Open Source and Free Software Licensing: Guide to Navigating Licensing Issues in Existing & New Software. [online] Google Books. 'O'Reilly Media, Inc.' Available at: https://books.google.gr/books?id=04jG7TTLujoC&pg=PT18&redir_esc=y#v=onepage&q&f=false
[Accessed 23 Feb. 2024].

The Strong National Museum of Play. (2011). Saving in Video Games. [online] Available at: https://www.museumofplay.org/blog/saving-in-video-games/

kb.iu.edu. (n.d.). What is the difference between a compiled and an interpreted program? [online] Available at: https://kb.iu.edu/d/agsz#:~:text=The%20difference%20between%20an%20interpreted

www.sdd-technology.com. (n.d.). Definition of the Term Cross-Platform or Hybrid App. [online] Available at: https://www.sdd-technology.com/news/definition-of-cross-platform

AtariAge Forums. (2006). 2600 games with an actual save feature? [online] Available at: https://forums.atariage.com/topic/88339-2600-games-with-an-actual-save-feature/
[Accessed 23 Feb. 2024]

Micheloni, R., Crippa, L. and Marelli, A. (2010). Inside NAND Flash Memories. [online] Google Books. Springer Science & Business Media. Available at: https://books.google.gr/books?id=vaq11vKwo_kC&pg=PA2&redir_esc=y#v=onepage&q&f=false
[Accessed 23 Feb. 2024].