ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ

ΣΧΟΛΗ ΟΙΚΟΝΟΜΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΚΑΙ ΔΙΟΙΚΗΣΗΣ ΕΠΙΧΕΙΡΗΣΕΩΝ

ΤΜΗΜΑ ΔΙΟΙΚΗΤΙΚΗΣ ΕΠΙΣΤΗΜΗΣ
ΚΑΙ ΤΕΧΝΟΛΟΓΙΑΣ

Τρισδιάστατες Πλατφόρμες Ανάπτυξης Παιχνιδιών. Μελέτη Περίπτωσης Δημιουργίας Διαδραστικής Εφαρμογής με τη Χρήση Πλατφόρμας Παιχνιδιών “Unity”

Πτυχιακή Εργασία

Πετρόπουλος Νικόλαος (ΑΜ:13043)

Επιβλέπων: Χαλκιόπουλος Κωνσταντίνος

ΠΑΤΡΑ 2020

Περίληψη

Η παρούσα πτυχιακή εργασία αφορά τις τρισδιάστατες πλατφόρμες δημιουργίας παιχνιδιών. Γίνεται μια σύντομη ιστορική διαδρομή των βιντεοπαιχνιδιών και αναλύονται τα χαρακτηριστικά και οι δυνατότητες μιας πλατφόρμας δημιουργίας βιντεοπαιχνιδιών. Αναλύουμε επιπλέον το κόστος , την ιστορία και την πολιτική τριών απο τις μεγαλύτερες εταιρίες σε αυτό τον χώρο.

Δημιουργήσαμε μια εφαρμογή / βιντεοπαιχνίδι δυο διαστάσεων (2D Puzzle Game) με την χρήση της «Unity» και τον προγραμματισμό σε γλώσσα C#. Έμπνευση για την δημιουργία μας έδωσε ενα ψυχολογικό φαινόμενο που παρατηρήθηκε απο τον John Ridley Stroop και ονομάστηκε Stroop Effect. Ασχολείται με τον χρόνο αντίδρασης αναγνώρισης λέξεων χρωμάτων, όταν αυτές αλλάζουν χρώμα γραμματοσειράς. Αναλύουμε αυτό το φαινόμενο απο δημοσιευμένα άρθρα βασισμένα σε αυτό το φαινόμενο.

Τέλος αφού παρουσιάσουμε και εξοικειώσουμε τον αναγνώστη της πτυχιακής με το περιβάλλον της Unity αλλά και το φαινόμενο Stroop, παρουσιάζουμε την εφαρμογή, αλλά και την υλοποίηση βήμα βήμα.

Λέξεις κλειδιά: Τρισδιάστατες Πλατφόρμες Δημιουργίας Βιντεοπαιχνιδιών, Unity, Φαινόμενο Στρούπ, Χρόνος αντίδρασης, Προγραμματισμός, C#

Abstract

This thesis is about 3D game developer platforms. After a brief introduction of video games history, we analyze the features and capabilities of game developer platforms. We further analyze the cost, history and policy of three of the largest companies in this field.

We created a 2D Puzzle Game using "Unity" and programming in C # language. Inspiration for creation gave us a psychological phenomenon observed by John Ridley Stroop and named Stroop Effect, after him. It deals with color-word recognition reaction time when they change font color. We analyze this phenomenon from published articles based on this phenomenon.

Finally, after introducing and familiarizing our reader with the Unity environment and the Stroop effect, we present the application and the implementation gradually.

Key Words: 3D Game developer platforms, Unity, C# Language, Stroop Effect, Color Word Recognition Reaction Time

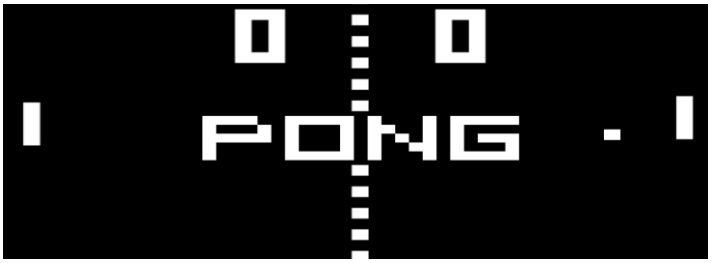
Περιεχόμενα

Πίνακας εικόνων

1 Εισαγωγή

1.1 Ιστορία των Βιντεοπαιχνιδιών

Εξ'ορισμού, ένα βιντεοπαιχνίδι είναι ένα ηλεκτρονικό παιχνίδι που περιλαμβάνει αλληλεπίδραση με μια διεπαφή του χρήστη, για τη δημιουργία οπτικής ανάδρασης σε μια συσκευή αναπαραγωγής εικόνας, όπως μια οθόνη τηλεόρασης ή οθόνη υπολογιστή. Το πρώτο παιχνίδι που δημιουργήθηκε ήταν το Tennis for Two σε οθόνη παλμογράφου, από τον William Higginbotham το 1958 και φτιάχτηκε με σκοπό την παρουσίασή του για τρεις ημέρες στο Brookhaven National Laboratory. Ύστερα άρχισαν να προγραμματίζονται βιντεοπαιχνίδια για εμπορικούς σκοπούς, μέσω ηλεκτρονικών μηχανών που λειτουργούσαν με κέρματα. Η πρώτη επιτυχία ήταν το Pong το 1972 που εκδόθηκε από την Atari .



Εικόνα 1 : Pong!

Έτσι άρχισε η χρυσή εποχή των ηλεκτρονικών μηχανών με τίτλους χαρακτηριστικούς όπως Breakout(1976), Space Invaders(1978), Pac-Man(1980), Donkey Kong(1981) και το Tetris(1984). Με την συνεχόμενη ανάπτυξη των υπολογιστών και την αύξηση των προσωπικών υπολογιστών (Personal Computers ή PC) όπως το Apple II και το IBM PC, άρχισαν και οι πρώτες κονσόλες βιντεοπαιχνιδιών, με παράδειγμα το Nintendo Entertainment System (ή NES) το 1985, όπου έκανε μεγάλη επιτυχία με τίτλους όπως Super Mario Bros.(1985), The Legend Of Zelda(1986) και Final Fantasy(1987), και η πρώτη φορητή κονσόλα Game Boy(1989) με χαρακτηριστικούς τίτλους Tetris(1989) και Pokémon Red & Blue(1996).

Το 1990 με την παραπάνω επεξεργαστική δύναμη των 16-bit και 32-bit των επεξεργαστών και με κάρτες γραφικών οι οποίες υποστήριζαν γραφικά τριών διαστάσεων (3D Graphics) και μοχλούς πάνω στο χειριστήριο που υποστήριζαν αναλογικό έλεγχο ακριβείας, φτιάχτηκαν κονσόλες όπως Sony PlayStation(1994), Nintento 64(1996) και επέτρεψαν τον πειραματισμό σε πολλά είδη βιντεοπαιχνιδιών και τίτλους που ήταν αρκετά καινοτόμοι. Συγκεκριμένα το GoldenEye 007(1997) αναφέρεται μέχρι και σήμερα ως ένα απο τα σπουδαιότερα παιχνίδια πρώτου

προσώπου (FPS), μαζί με το The Legend of Zelda: Ocarina of Time(1998) που θεωρείτε και αυτό ένα από τα καλύτερα παιχνίδια όλων τον εποχών.

Εικόνα 2 : Αριστερά το εξώφυλο του Goldeneye007 ενώ αριστερά στιγμιότυπο απο το παιχνίδι

Στις αρχές του 2000 κυκλοφόρησαν παιχνίδια που επηρέασαν ολόκληρα gaming genres (είδη βιντεοπαιχνιδιών). Το Grand Theft Auto 3 (2001) όπου ήταν από τα πιο δημοφιλή Sandbox παιχνίδια. Sandbox θεωρείτε ένα στυλ παιχνιδιού το οποίο αφήνει τον παίχτη ελεύθερο σε όλο τον χάρτη χωρίς περιορισμούς (open world), δηλαδή χωρίς επίπεδα (levels), δηλαδή είτε να αλλάξει τον χάρτη ή το περιβάλλον του χαρακτήρα με δικιά του βούληση. Την ίδια χρονιά η Microsoft μπαίνει στην βιομηχανία παιχνιδιών λανσάροντας την δικιά της παιχνιδομηχανή Xbox, μαζί με την αποκλειστικότητα του Halo franchise (2001) και το καινούργιο PlayStation 2 (2000). Βασισμένο στο παιχνίδι Warcraft 3 (2002) κυκλοφορεί ως modification(εν συντομία mod και σημαίνει την μετατροπή ενός βιντεοπαιχνιδιού ως το προς συμπεριφέρεται και πως δείχνει) το Defense of The Ancients(2003) και δημιουργεί το δικό του game genre, που είναι ακόμα ένα από τα πιο δημοφιλές genre, το MOBA(Multiplayer Online Battle Arena). Μεγάλη επιρροή στα MMORPG(Massive Multiplayer Online Role Playing Game) δημιούργησε το World Of Warcraft(2004) αφού το 2010 έφτασε τους 12 εκατομμύρια συνδρομητές δίνοντάς του τον τίτλο ως το πιο δημοφιλές MMORPG από το Guinness World Record. Η Valve λανσάρει την 3D μηχανή βιντεοπαιχνιδιού Source (2004) και μαζί παιχνίδια βασισμένα στη μηχανή όπως το Half life 2 (2004) και modifications του Half Life 2 έγιναν επιτυχημένα παιχνίδια σαν το Counter Strike:Source (2004), Portal (2007) και Team Fortress 2 (2007).

Εικόνα 3 : Αριστερά το Grand Theft Auto 3, στη μέση το Half Life 2 και στα δεξιά το DotA

Το 2006 ήταν η χρονιά που η Nintendo επιχειρεί κάτι καινοτόμο, λανσάροντας το Wii μαζί με το Wii Sports (2006), αφήνοντας πίσω το παραδοσιακό χειριστήριο και επιτρέπει στον χρήστη, σύμφωνα με τις κινήσεις που κάνει, να χειρίζεται την κονσόλα και να προσομοιώνει αθλήματα, όπως τένις και γκολφ, βάση του χειριστηρίου με σένσορες κίνησης. Είχε αποτέλεσμα να αναπτυχθεί μια καινούργια τεχνολογία, την οποία χρησιμοποιούμε μέχρι και σήμερα ειδικά σε VR(Virtual Reality) περιβάλλοντα.

Ίδια περίοδος με το Wii, κυκλοφορούν οι απαντήσεις της Sony και της Microsoft, με το PlayStation 3 και Xbox 360 αντίστοιχα. Και οι τρεις εταιρίες εκμεταλλεύονται την ανάπτυξη του internet και δημιουργούν ένα δίκτυο διασύνδεσης των δικών τους κονσόλων και ένα online καταστήματα (Xbox live για το Xbox 360, PlayStation Network για το Playstation 3, Wii Shop Channel για το Wii).

Εικόνα 4 : Αριστερά το Playstation 3, στη μέση το Xbox 360 και στα δεξιά το Wii

Με την μεγάλη επιτυχία του Facebook, άρχισαν να αναπτύσσονται και τα βιντεοπαιχνίδια με βάση πλατφόρμες κοινωνικής δικτύωσης όπως το Farmville (2009) τα οποία έχουν μικρότερο χρόνο gameplay (αλληλεπίδραση με τον χρήστη) αλλά χρησιμοποιεί άλλες τεχνικές για να κρατήσουν το κοινό τους, όπως ανταμοιβές (παράσημα, τρόπαια και επιτεύγματα), δώρα σε φίλους που έχουν στα μέσα κοινωνικής δικτύωσης, ανταγωνισμός μεταξύ τους, αλλά και την συνεργασία τους. Σαν αποτέλεσμα να εδραιωθεί η λειτουργία του gamification, δηλαδή παιχνιδοποίηση διάφορων περιβαλλόντων για την προσέλκυση ατόμων καθώς και την αύξηση συμμετοχής τους, σε πολλές υπηρεσίες και sites σαν το Samsung Nation (2011) και Foursquare (2009).

Φτάνοντας στο 2010 κυκλοφορούν η όγδοης γενιάς κονσόλες. Η Nintendo κυκλοφορεί το Nintendo 3DS (2011) και το Wii U (2011), η Sony κυκλοφορεί το PlayStation Vita(2011) και το PlayStation 4(2013) και η Microsoft το Xbox One(2013).

Σήμερα έχουν κυκλοφορήσει, οι βελτιωμένες εκδόσεις, Playstation 4 Pro (2016), Xbox One X (2017), Nintendo Switch (2017). Και το Playstation 4 Pro και το Xbox One X είναι οι βελτιώσεις εκδόσεις των προκατόχων τους για την υποστήριξη 4K gaming, κάνοντάς τα πολύ δυνατές παιχνιδομηχανές. Η Nintendo πάλι καινοτόμησε, φτιάχνοντας μια φορητή συσκευή η οποία μπορεί να συνδεθεί με μία βάση, ενεργοποιώντας ολη την ισχύ της μηχανής, φτάνοντας τα 1080p με 60 fps (frames per second).

Εικόνα 5 : Αριστερά το Nintento Switch, στη μέση το Playstation 4 και στα δεξιά το Xbox One

Πλέον οι απαιτήσεις των βιντεοπαιχνιδιών, μιας και η αγορά έχει μεγάλο υποστηρικτικό κοινό ειδικά με την προσθήκη των e-sports (διαγωνισμοί μεμονωμένων επαγγελματικών παιχτών ή ομάδων γύρω από βιντεοπαιχνίδια), έχουν ανέβει. Ως αποτέλεσμα η επεξεργαστική ισχύ των προσωπικών υπολογιστών και των κονσόλων να χρειάζεται να προσαρμοστεί σε αυτά. Το PlayStation 4 Pro έχει δύο τετραπύρηνους επεξεργαστές στα 2.13GHz, με GPU (graphic processing unit) AMD Radeon GCN στα 911 MHz και 8gb RAM. Το Xbox One διαθέτει οκταπύρηνο επεξεργαστή στα 1.75GHz,με GPU 853MHz kai 8gb RAM. Ενώ ένας υπολογιστής μπορεί να φτάσει τις ακραίες αποδόσεις των 4.2GHz με τετραπύρινο επεξεργαστή,GPU 11GB στα 1531MHz και τέλος 32gb RAM .

2 Game Engines

2.1 Βασικά Χαρακτηριστικά μιας μηχανής

Μια μηχανή παιχνιδιού είναι ένα σύστημα λογισμικού σχεδιασμένο για τη δημιουργία και την ανάπτυξη βιντεοπαιχνιδιών. Κάνει όλες τις απαραίτητες διεργασίες και έχει τα απαραίτητα εργαλεία για την δημιουργία βιντεοπαιχνιδιών. Χρησιμοποιούνται από εταιρίες ή για προσωπική χρήση, διότι δεν χρειάζεται να φτιάξεις από την αρχή αυτά τα βασικά εργαλεία όπως είναι η βαρύτητα και πως αλληλοεπιδρούν διάφορα αντικείμενα με αυτήν, και επικεντρώνονται στην ιστορία, τους χαρακτήρες και γενικά υλοποιούν την ιδέα τους χωρίς να σπαταλήσουν επιπλέον χρόνο και λεφτά.

Τα βασικά εργαλεία για τις μηχανές βιντεοπαιχνιδιών είναι:

1) Μηχανή εξαγωγής γραφικών (Rendering Engine)

Χρησιμοποιώντας τα 3D μοντέλα σε διάφορες μορφές αρχείων, που έχουμε αγοράσει έτοιμα ή έχουμε δημιουργήσει με προγράμματα όπως Blender και Maya, επεξεργάζεται αυτά τα αρχεία και το μεταφράζει σε ένα αντικείμενο μέσα στο περιβάλλον του παιχνιδιού μέσω API (Application Programming Interface ή Διεπαφή προγραμματισμού εφαρμογών) όπως το OpenGL, Directx, Vulkan ή nVidia VRworks. Επίσης όταν δημιουργήσουμε τον κόσμο χειρίζεται τα φώτα (ακτίνες του ήλιου , διαθλάσεις και πως αντιδρά το φως πάνω ή μέσα από τα αντικείμενα αναλόγως το υλικό τους), τις σκιές καθώς και διάφορες κινήσεις των μοντέλων.

2) Μηχανή ήχου (Sound Engine)

Μηχανή που αναπαράγει αρχεία ήχου μέσα σε ένα 3D περιβάλλον και πως αντιδρά με τον χαρακτήρα και την απόστασή του ανάμεσα στον χαρακτήρα και την πηγή του ήχου. Μπορεί να προσαρμόσει το αρχείου ήχου προσθέτοντας εφέ όπως την ηχώ, το φαινόμενο Doppler και εξισορρόπηση ήχου.

3) Μηχανή Φυσικής (Physics Engine)

Η κύρια λειτουργεία της μηχανής φυσικής είναι η προσομοίωση των νόμων της φυσικής. Για παράδειγμα ανιχνεύει συγκρούσεις αντικειμένων και αντιδράσεων διαφορετικών σχημάτων μεταξύ τους, μέτρηση αποστάσεων, υπολογισμό ταχυτήτων, δυναμική των υγρών και αερίων.

4) Μηχανή Τεχνητής Νοημοσύνης (Artificial Intelligence)

Τα βιντεοπαιχνίδια χρησιμοποιούν συνήθως NPCs (Non-player Characters), δηλαδή χαρακτήρες που δεν χειρίζονται απο παίχτες και αλληλεπηδρούν μαζί τους. Έχουν συμπεριφορές όπως μια επίθεση στον χαρακτήρα μας (εχθρός) ή να μας διηγηθεί ένα μέρος της ιστορίας με διαδραστικούς διαλόγους ή την κίνηση των αντικειμένων μέσα

στον χώρο (Pathfinding). Αυτές τις συμπεριφορές καλείτε να τις δώσει ο προγραμματιστής και η μηχανή μέσω μιας ρουτίνας διεργασιών για να τις αποδώσει στον έκαστο χαρακτήρα.

5) Μηχανή Αναπαράστασης Διεπαφής Χρήστη (Graphical User Interface)

Οι περισσότερες μηχανές πλέον έχουν μηχανισμούς για να κρατάνε βαθμολογία, να εμφανίζουν παράθυρα διαλόγων και να δημιουργούν μενού στα οποία επιλέγεις βασικές λειτουργίες του προγράμματος, όπως κουμπιά, στοίχιση και ευθυγράμμιση αντικειμένων στο περιβάλλον διεπαφής χρήστη και συμπεριφορά αυτών των αντικειμένων με εφέ.

2.2 Γνωστές Μηχανές Βιντεοπαιχνιδιών

Υπάρχουν αρκετές μηχανές βιντεοπαιχνιδιών οι οποίες έχουν διάφορα συστήματα και καινοτομίες. Για την σωστή επιλογή μιας μηχανής θα πρέπει να δώσουμε βάση σε πολλούς παράγοντες διότι πέρα από τα εργαλεία που μας παρέχουν, πρέπει να γνωρίζουμε και τα πνευματικά δικαιώματα και τις άδειες καθώς και το κόστος τους, που έχουν για την χρήση τους για εμπορικούς σκοπούς. Η αγορά των μηχανών είναι μεγάλη αλλά θα αναλύσουμε τις δημοφιλέστερες επιλογές, από πλευρά μεγάλων εταιριών αλλά και μικρών πρότζεκτ.

Id Tech

Είναι η πρώτη μηχανή βιντεοπαιχνιδιών που κυκλοφόρησε δημόσια. Αναπτύχθηκε από τέσσερις προγραμματιστές στο Dallas, με τον John Carmack να γράφει τον βασικό κώδικα. Ξεκίνησε αρχικά από την ανάπτυξη των πρώτων 3D παιχνιδιών με χαρακτηριστικό παιχνίδι της εποχής να είναι το Wolfenstein. Η Id Softwares πούλησε παράλληλα με το Doom τον πηγαίο κώδικα του παιχνιδιού στο κοινό. Γνωστά παιχνίδια βασίστηκαν πάνω στην μηχανή όπως Wolfenstein franchise, Doom franchise, Quake franchise και Rage. Αυτή την στιγμή η Id Tech 6 χαρακτηρίζεται σαν ιδιόκτητο λογισμικό, δηλαδή η εταιρίες που το χρησιμοποιούν έχουν όλα τα πνευματικά δικαιώματα, και δεν είναι διαθέσιμο για ιδιώτες πάρα μόνο σε συγκεκριμένα στούντιο.

Εικόνα 6 : Το logo της IdTech

Unreal Engine

Η Unreal Engine είναι μια παιχνιδομηχανή που έχει αναπτυχθεί από την Epic Games. Παρουσιάστηκε με το Unreal(1998) και δέχτηκε πολλές επιρροές από την id tech η οποία είχε δημιουργηθεί πριν λίγα χρόνια. Σήμερα η Unreal Engine έχει χαρακτηριστεί ως η πιο επιτυχημένη παιχνιδομηχανή, διότι έχει χρησιμοποιηθεί για πολλά και επιτυχημένα παιχνίδια (Λίστα βιντεοπαιχνιδιών που χρησιμοποιούν Unreal Engine). Πλέον με την Unreal Engine 4 να είναι διαθέσιμη δωρεάν, χρησιμοποιείται πειραματικά από εταιρίες ή ιδιώτες για την υλοποίηση της ιδέας τους. Εάν όμως γίνει δημοσίευση της εφαρμογής/παιχνιδιού και ξεπεράσει τα 3,000$ κέρδη ανά τετράμηνο, οφείλουν στην Epic ένα ποσοστό ύψους 5% κάθε τετράμηνο από τα κέρδη τους. Αυτό μπορεί να αλλάξει με προσαρμοσμένη άδεια κατόπιν συνεννόησης με την Epic. Η γλώσσα προγραμματισμού που χρησιμοποιεί είναι η C++, με ορισμένες διαφοροποιήσεις. Μέσω του Unreal Marketplace οι χρήστες μπορούν να αγοράζουν, να πουλούν ή ακόμα και να μοιράζονται assets (περιουσιακά ψηφιακά στοιχεία), δηλαδή έτοιμα "πράγματα" που χρησιμοποιούνται στο παιχνίδι όπως ήχοι και μουσικές, μοντέλα και υφές υλικών (textures).

Εικόνα 7 : Το logo της Unreal Engine

Unity

Η Unity είναι μια παιχνιδομηχανή η οποία δημιουργήθηκε το 2005. Κυκλοφόρησε εξ ολοκλήρου για Mac OS X στην αρχή. Πλέον κυκλοφορεί και για Windows OS. Οι γλώσσες προγραμματισμού που χρησιμοποιεί είναι η C# και Javascript. Χρησιμοποιείτε και σε 3D και σε 2D περιβάλλον, μαζί με τις αντίστοιχες μηχανές φυσικής nVidia, physX και Box2D. Ο μεταγλωτιστής (compiler) που έχει, μετρατρέπει τον κώδικα σε αρκετές πλατφόρμες παραδείγματος χάρη, Android, Windows, Xbox, Facebook χωρίς να χρειάζεται απο τον προγραμματιστή αλλαγή στον κώδικα για την μεταφορά της εφαρμογής σε άλλη πλατφόρμα. Διατίθεται δωρεάν (Unity Personal) για προσωπική χρήση και κυκλοφορία βιντεοπαιχνιδιού / εφαρμογής μέχρι τα 100.000$ ετησίως. Από εκεί και πέρα χρειάζεται αναβάθμιση στο Unity Plus με μια συνδρομή 35$/μήνα. Εάν τα κέρδη ξεπεράσουν και τα 200.000$ θα πρέπει να αναβαθμίσουμε το λογαριασμό σε Unity Pro με 125$/μήνα. Οι χρεώσεις αφορούν κάθε λογαριασμό Unity ξεχωριστά και για ένα άτομο. Και σε αυτήν την παιχνιδομηχανή

υπάρχει το Unity Asset Store που επιτρέπει την αγοραπωλησία και ανταλλαγή asset. Επειδή διατίθεται δωρεάν την κάνει διαδεδομένη σε χρήστες που δημιουργούν βιντεοπαιχνίδια για χόμπι, για εκπαιδευτικούς λόγους αλλά και για μικρά και πειραματικά studios. Αυτό δεν αποτρέπει μεγάλες και γνωστές εταιρίες ΑΑΑ (ορολογία για τεράστια και ακριβά studios) να χρησιμοποιούν την Unity. Μερικά παραδείγματα γνωστών βιντεοπαιχνιδιών.

Εικόνα 8 : Το logo της Unity

3 Stroop Effect

3.1 Εισαγωγή

Το Stroop Effect πήρε το όνομά του απο τον John Ridley Stroop. Ο John Ridley Stroop ήταν ένας Αμερικάνος νευροψυχολόγος που δημοσίευσε την θεωρία του με τίτλο “Studies of Interference in Serial Verbal Reactions" στο Journal of Experimental Psychology,Vol 18, 643–662 το 1935. Η θεωρία του υποστήριζε πως υπάρχει παρεμβολή στον χρόνο αντίδρασης μιας διαδικασίας. Συγκεκριμένα, βάση των πειραμάτων του, το χρώμα της λέξης δεν αντιστοιχεί στην σημασιολογική έννοιά της (π.χ. Κόκκινο) και παρουσιάζει καθυστέρηση με την περίπτωση που υπήρχε αντιστοιχία (π.χ. Κόκκινο) στην αναγνώρισή της. Καθώς και ότι η συνεχόμενη εξάσκηση πάνω σε αυτό το φαινόμενο, θα μείωνε το χρόνο που χρειάζεται κάποιος για την αναγνώρισή της στην πρώτη περίπτωση.

Ο Stroop δεν ήταν ο πρώτος που ασχολήθηκε με αυτή την θεωρία. Ο McKeen Cattell ασχολήθηκε με αυτή την θεωρία (Cattell, J.M.: The time it takes to see and name objects. Mind 11, 63–65 (1886)) και αναφέρει ότι η ανάγνωση είναι μια αυτόματη διαδικασία, που την εξασκούμε από πολύ μικρή ηλικία, δηλαδή κάτι μη ελεγχόμενο, μη θελημένο και γρήγορο. Δεν γίνεται να «απενεργοποιήσουμε» αυτή την λειτουργία και γι’ αυτό παρεμβάλει με την προσπάθεια να ονομάσουμε το χρώμα της γραμματοσειράς. Το ίδιο είπε και ο Peterson (1925) εξηγώντας ότι οι λέξεις προκαλούσαν μία μόνο απόκριση απάντησης, ενώ τα χρώματα πολλαπλές, εξού και η καθυστέρηση.

Ύστερα από περίπου 50 χρόνια(1991) ο Colin M. MacLeod εκδίδει το «Half a Century of Research on the Stroop Effect: An Integrative Review», μελετώντας, συγκεντρώνοντας και οργανώνοντας περίπου 700 άρθρα σχετικά με την μελέτη του Stroop. Καθώς και νέα ευρήματα από έρευνες και πειράματα με τον Dunbar(1988). Ο ίδιος υποστηρίζει μέσα από το άρθρο του, όσο περισσότερο εξασκείται μια εκδοχή του πειράματος του Stroop, τόσο περισσότερο δημιουργεί προβλήματα σε άλλες εκδοχές.

3.2 Τα τρία βασικά πειράματα του Stroop

Μέσα από το “Studies of Interference in Serial Verbal Reactions" ο Stroop εξήγησε τρείς διαδικασίες-πειράματα, που τον βοήθησαν να καταλήξει στην θεωρία του. Αρχικά τα χρώματα που χρησιμοποίησε ήταν το κόκκινο, το μπλε, το πράσινο, το καφέ και το μωβ. Τα χρώματα ρυθμίστηκαν με βάση τρία πράγματα. Πρώτον, κανένα χρώμα δεν είχε το ίδιο χρώμα γραμματοσειράς με την λέξη του χρώματος, αλλά επαναλαμβανόταν ίσες φορές με τα άλλα τέσσερα χρώματα γραμματοσειράς. Δεύτερον, ρυθμίστηκαν τα χρώματα γραμματοσειράς και οι λέξεις χρώματος, με τέτοιο τρόπο ώστε να αποφευχθεί η πιθανότητα διαδοχικής εμφάνισης σε γραμμή ή στήλη. Τρίτον, κάθε χρώμα γραμματοσειράς και κάθε λέξη χρώματος εμφανίζεται συνολικά δύο φορές.

Το πρώτο πείραμα του Stroop είχε δύο σκέλη. Το πρώτο σκέλος αφορούσε την ανάγνωση δυνατά των λέξεων που εμφανιζόντουσαν όταν οι λέξεις ήταν διαφορετικό χρώμα (RCNd όπως το ονόμασε ο ίδιος) και το δεύτερο σκέλος την αντίστροφη διαδικασία, δηλαδή αφορούσε την ανάγνωση των λέξεων όταν το χρώμα γραμματοσειράς ήταν μαύρο (RCNb). Το πείραμα είχε δείγμα 70 ατόμων και το αποτέλεσμά του ήταν διαφορά (D) μόνο 2,3 δευτερολέπτων ανάμεσα στο RCNd και το RCNb, όπως φαίνεται και στην παρακάτω εικόνα.

TABLE I

THE MEAN TIME IN SECONDS FOR READING ONE HUNDRED NAMES OF COLORS PRINTED IN COLORS DIFFERENT FROM THAT NAMED BY THE WORD AND FOR ONE HUNDRED NAMES OF COLORS PRINTED IN BLACK

Sex	No. Ss.	*RCNd*	σ	*RCNb*	σ	*D*	PE_d	D/PE_d
Male	14	43.20	4.98	40.81	4.97	2.41	1.27	1.89
Female	56	43.32	6.42	41.04	4.78	2.28	.72	3.16
Male and Female	70	43.30	6.15	41.00	4.84	2.30	.63	3.64

Εικόνα 9 : Αποτελέσματα πρώτου πειράματος του Stroop

Το δεύτερο πείραμα δημιουργήθηκε από το πρώτο σκέλος (RCNd) με την ίδια σειρά, αλλά αντί για λέξεις χρησιμοποίησε γεμάτα τετράγωνα(□) και το ονόμασε NC (Naming color test). Η διαδικασία του δεύτερου πειράματος ήταν να γίνει αναφορά του χρώματος της γραμματοσειράς που ήταν γραμμένη η λέξη ("Naming color or word test where the color of the print and the word are different" ή εν συντομία NCWd), συγκρίνοντας και τα αποτελέσματα του NC. Το πείραμα είχε δείγμα 100 ατόμων και το αποτέλεσμα του ήταν η διαφορά (D) των 47 δευτερολέπτων ανάμεσα στο NC και στο NCWd. Ήταν μια αύξηση 74% στον χρόνο που χρειάστηκαν για να τελειώσουν το πείραμα. Η εικόνα (Εικόνα 10) μας εμφανίζει αυτά τα αποτελέσματα.

TABLE III

THE MEAN TIME FOR NAMING ONE HUNDRED COLORS PRESENTED IN SQUARES AND IN THE PRINT OF WORDS WHICH NAME OTHER COLORS

Sex	No. Ss.	*NCWd*	σ	*NC*	σ	*D/NC*	*D*	PE_d	D/PE_d
Male	29	111.1	21.6	69.2	10.8	.61	42.9	3.00	13.83
Female	71	107.5	17.3	61.0	10.5	.76	46.5	1.62	28.81
Male and Female	100	110.3	18.8	63.3	10.8	.74	47.0	1.50	31.38

Εικόνα 10 : Αποτελέσματα δεύτερου πειράματος του Stroop

Για το τρίτο πείραμα ο Stroop άλλαξε το σύμβολο του NC test από το γεμάτο τετράγωνο(□) σε σβάστικα (卍), διότι σύμφωνα με τον ίδιο ήταν ένα κοντινό σύμβολο με τους αλφαριθμητικούς χαρακτήρες γιατί εμπεριέχει και το λευκό χρώμα ενδιάμεσα απο τις γραμμές. Η διαδικασία του τρίτου πειράματος ήταν έκθεση των 32 εθελοντών στα προηγούμενα πειράματα (RCNb, RCNd , NC, NCWd) για 14 συνεχόμενες μέρες, εκτός από τις ενδιάμεσες μέρες 3 και 4 όπως 8 και 9 όπου δεν έγιναν τα πειράματα, όπως δείχνει και η παρακάτω εικόνα.

Day	1	2	3	4	5	6	7
Test	*RCNb*	*RCNd*	*NC*	*NCWd*	*NCWd*	*NCWd*	*NCWd*
Day	8	9	10	11	12	13	14
Test	*NCWd*	*NCWd*	*NCWd*	*NCWd*	*NC*	*RCNd*	*RCNd*

Εικόνα 11 : Πλάνο τρίτου πειράματος

Σκοπός ήταν να μάθει εάν υπάρχει καμπύλη βελτίωσης στα αποτελέσματα των εθελοντών από την πρώτη ημέρα μέχρι την τελευταία. Τα αποτελέσματα του τρίτου πειράματος έδειξαν μια φυσιολογική καμπύλη βελτίωσης, υποστηρίζοντας την θεωρία του Peterson ότι ”τα υποκείμενα μιας φυσιολογικής ετερογένειας θα εξειδικευτούν ευκολότερα σε απλές διαδικασίες ή δραστηριότητες, σχέση με πιο σύνθετες διαδικασίες” (ελεύθερη μετάφραση από Peterson and Barlow,1928, p. 228). Δηλαδή για τα πειράματα NCWd και NC, οι εθελοντές έδειξαν βελτίωση 16.8 δευτερολέπτων και 4 δευτερολέπτων αντίστοιχα, ενώ για το πείραμα RCNd μείωση 15.4 δευτερολέπτων από τα αρχικά αποτελέσματα της πρώτης μέρας. Τα αποτελέσματα φαίνονται στην παρακάτω εικόνα.

TABLE VI

A Summary of the Means in Table V, Showing the Effect of Practice in the *NCWd* Test upon the *NCWd*, the *NC*, and the *RCNd* Tests

Test	*NCWd*			*NC*			*RCNd*		
Sex	M	F	M & F	M	F	M & F	M	F	M & F
Initial Score	51.2	47.8	49.6	30.6	26.5	28.7	19.6	19.1	19.4
Final Score	33.4	31.5	32.8	25.9	23.6	24.7	37.3	32.0	34.8
Gain	17.8	16.3	16.8	4.7	2.9	4.0	−17.7	−12.9	−15.4
Percent Gain	34.8	34.1	33.9	15.4	10.9	13.9	−90.3	−67.5	−79.3

Minus sign shows loss

Εικόνα 12 : Αποτελέσματα τρίτου πειράματος του Stroop

3.3 Παραλλαγές των αρχικών πειραμάτων του Stroop

Παραλλαγές των αρχικών πειραμάτων του Stroop έγιναν πριν και μετά την θεωρία του. Παρακάτω αναφέρονται κατηγορίες των παραλλαγών, όπως τις ομαδοποίησε ο McLeon στο άρθρο του "Half a Century of Research on the Stroop Effect: An Integrative Review" (1991).

1) Διαφορετική Βαθμολόγηση του Τεστ

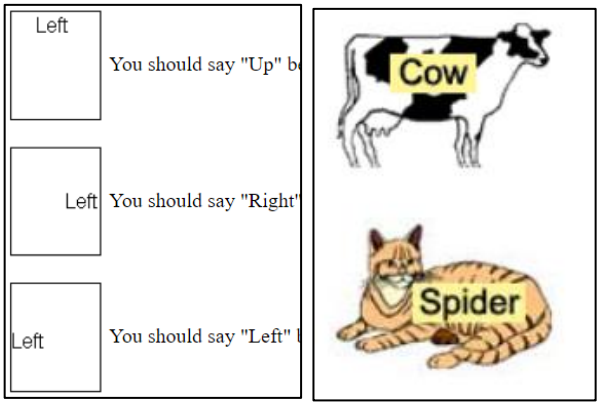
Μικρές τροποποιήσεις του πρώτου πειράματος έγιναν απο πολλούς επιστήμονες, έτσι ώστε να πάρουν αποτελέσματα σε συγκεκριμένους τομείς της ψυχολογίας. Τροποποιήσεις διαφέρουν στα υλικά του τέστ, την χορήγηση του τέστ, το χρόνο χορήγησης, καθώς και στο σκορ. Οι Thurstone and Mellinger (1953) τροποποίησαν το αρχικό πείραμα βάζοντας μαύρο φόντο αντί για άσπρο και χρησιμοποίησαν τέσσερα χρώματα. Ο Jensen(1956) προσπάθησε να μεγαλώσει τις κάρτες-ερεθίσματα για να μην υπάρχει καθυστέρηση στην αλλαγή καρτών. Οι κάρτες C(color) ήταν 10x10(στήλες x σειρές), ενώ οι κάρτες W και CW 5x25.

2) Ταξινόμηση και αντιστοίχιση

Σε αυτή την παραλλαγή, ο εξεταζόμενος πρέπει να αντιστοιχίσει σε κατηγορίες τις κάρτες ερεθίσματος. Πιο συγκεκριμένα, εμφανίζονται δύο εικόνες/σχήματα σε δύο χρώματα. Η πρώτη φάση είναι η αντιστοίχηση των καρτών στις σωστές κατηγορίες σύμφωνα με το σχήμα τους, ενώ η δεύτερη φάση (αντίστροφη διαδικασία Stroop) είναι η αντιστοίχηση των καρτών στις κατηγορίες σύμφωνα με το χρώμα τους.

3) Διαδικασία Εικόνας-Λέξης

Απο τις πρώτες θεωρίες για το φαινόμενο Stroop, παρατηρήθηκε καθυστέρηση και όταν υπάρχουν εικόνες και λέξεις. Συγκεκριμένα λέξεις εμφανίζονταν μέσα σε ζωγραφιές. Παρατηρήθηκε μεγαλύτερη καθυστέρηση στην διαδικασία όταν οι λέξεις είχαν άμεση σχέση με την εικόνα και δεν ήταν μια ανεξάρτητη λέξη. Η εικόνα μπορεί να είναι και μια λέξη ή πολλές ίδιες λέξης που λειτουργούν σαν εικόνα. Αυτή η διαδικασία πραγματοποιείται και με βέλη με φορά αριστερά-δεξιά με τον εξεταζόμενο να αναγνωρίζει την φορά του βέλους. Αλλά και με γεωμετρικά σύμβολα όπου



Εικόνα 13 : Παραλλαγή Πειράματος με την διαδικασία Εικόνας-Λεξης

εμφανιζόταν η λέξη και ύστερα έπρεπε να επιλέξουν το σωστό γεωμετρικό σχήμα. Κάποια άλλα παραδείγματα φαίνονται στην εικόνα 13.

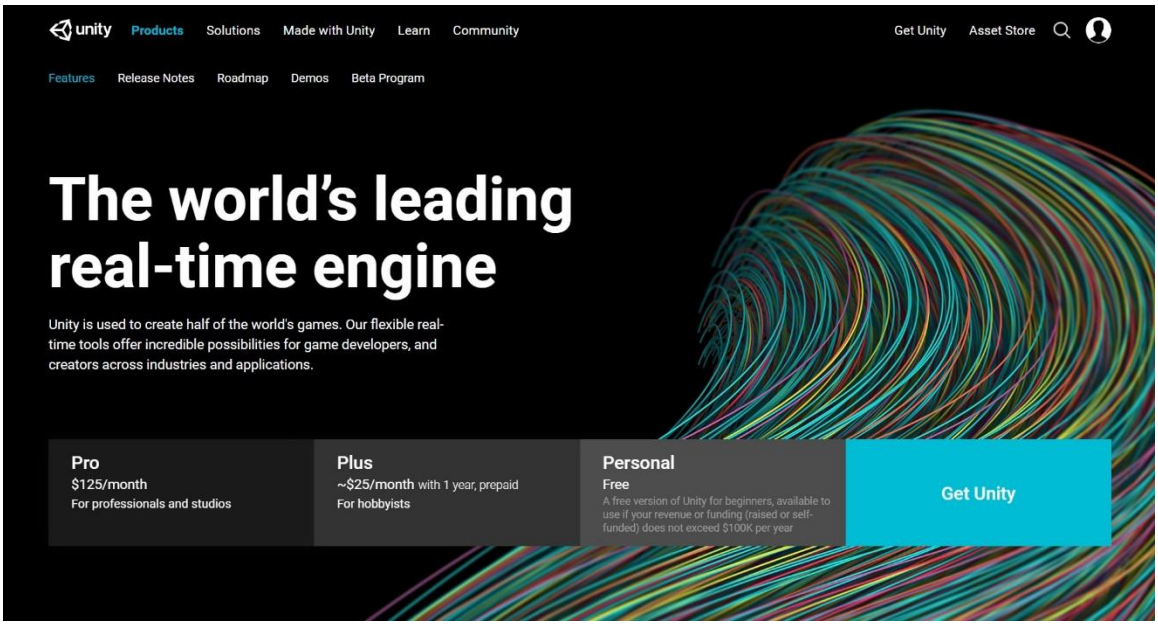
4) Ηχητική διαδικασία Stroop

Μιά παραλλαγή η οποία εξετάζει την αντίδραση και την παρεμβολή σε ήχους. Μια διαδικασία είναι η αναπαραγωγή ήχου χαμηλών ή υψηλών συχνοτήτων, και την αντίδραση απο τον εξεταζόμενο στην αναγνώρησή τους. Άλλη διαδικασία είναι η αναγνώρηση του φύλου (άνδρας ή γυναίκα), όταν προφέρουν λέξεις. Η διαδικασία μπορεί να πραγματοποιηθεί και με αναγνώρηση της ταχύτητας, της έντασης και της διάρκειας των λέξεων που ακούνε.

4 Unity

4.1 Εισαγωγή στην Unity

Η Unity είναι ενα εργαλείο για προγραμματιστές, καλλιτέχνες αλλά και για σχεδιαστές. Μπορεί να χρησιμοποιηθεί για παιχνίδια, κινούμενα βίντεο και εφέ, εικονική πραγματικότητα (VR), αλλά και για επαυξημένη πραγματικότητα (AR). Χρησιμοποιείτε απο μεγάλα και μικρά studios και από πολυεθνικές σε διαφορους τομείς της τεχνολογίας όπως Windows, Apple, Android, Xbox, Nintendo, Playstation, Steam, Facebook, Toyota, Audi, Disney.



Εικόνα 14 : Αρχική ιστοσελίδα της Unity

Η πρώτη επαφή με την Unity γίνεται μέσω του ιστότοπού της. Η ηλεκτρονική διεύθυνσή της είναι « https://unity3d.com/ » . Στην αρχική σελίδα σαν επιλογές έχει:

Products

Σε αυτή την εννότητα, βλέπουμε κάποια απο τα χαρακτηριστικά της Unity και τι μπορεί να προσφέρει ο editor της. Με την εννότητα «Explore Unity» μας εξηγεί γιατί αποτελεί μια απο της επιτυχημένες μηχανές παιχνιδιών αλλά και όχι μόνο. Αν περιηγηθεί κανείς και στις αλλες εννότητες μπορεί να μάθει για όλες τις προηγούμενες ενημερώσεις (Release notes), το όραμα και το μέλλον για την Unity (Roadmap), την εγκατάσταση δοκιμαστικής έκδοση (demo) κάποιων προϊόντων με την χρήση της Unity και τέλος την εγγραφή στην δοκιμαστική έκδοση της Unity, όπου μπορείς να δοκιμάσεις εάν η εφαρμογή σου παρουσιάζει προβλήματα στην καινούργια έκδοση,

αλλαγές στην επερχόμενη έκδοση, καθώς και ανατροφοδότηση για την καινούργια έκδοση μέσω της κοινώτητάς της.

Solution

Στην συγκεκριμένη εννότητα, παρακολουθούμε πως μπορούμε να χρησιμοποιήσουμε την Unity. Όπως προαναφέρηκε, δεν περιορίζεται μόνο στην κατασκευή παιχνιδιών. Πολλοί σχεδιαστές κατασκευάζουν ολόκληρες ταινίες και ταινίες μικρούς μήκους. Μαθαίνουμε επίσης οτι και εταιρίες κατασκευής αυτοκινήτων χρησιμοποιούν την Unity για την κατασκευή ανταλλακτικών, καθώς και την προσομοίωση ενός καινούργιου αυτοκινήτου, πριν καν κατασκευαστεί. Μπορούν να εκμεταλευτούν τη μηχανή φυσικής και να δημιουργήσουν ενα ιδανικό περιβάλλον, χωρίς περιορισμούς απο την φύση και να δοκιμάσουν νεα προϊόντα. Η Unity μπορεί να προσομοιώσει και σχέδια αρχιτεκτονικής, μηχανικών και κατασκευαστών με μια εικονική μακέτα, μειώνοντας το κόστος και το χρόνος κατασκευής της μακέτας. Παράλληλα, εταιρίες την χρησιμοποιούν και για κατασκευή τυχερών παιχνιδιών, καθώς και για δημιουργία εικονικής και επαυξημένης πραγματικότητας.

Made with unity

Με το πάτημα σε αυτό τον σύνδεσμο, μπορεί κανείς να μάθει για τις επιτυχημένες εφαρμογές βασισμένες στην Unity, καθώς και τις ιστορίες πίσω απο αυτές τις εφαρμογές απο τους ίδιους τους προγραμματιστές και τις δυσκολίες που αντιμετώπισαν μέχρι την επιτυχία τους. Κάποιες απο αυτές είναι το Hollow Knight (Platform), Cuphead (Platform), Escape from Tarkov (Survival FPS), Rick and Morty:Virtual Rick-ality (VR βασισμένο στο πετυχημένο animation Rick and Morty), Sonder (Animated ταινία μικρού μήκους), Trinity (Ταινία VR). Όπως βλέπουμε και στον ιστότοπο, η γκάμα προϊόντων είναι αρκετά μεγάλη.

Learn

Αυτή η εννότητα είναι αφιερωμένη στην εκμάθηση της Unity. Χωρίς να υπάρχει καμία προαπαιτούμενη γλώσσα ή εμπειρία στον τομέα, μπορεί οποιοσδήποτε με αυτή την εννότητα να χρησιμοποιήσει την Unity όπως επιθυμεί. Βάση μαθημάτων, οδηγών, ζωντανών μεταδώσεων μαθημάτων, της ανταλλαγής πληροφοριών και βοήθειας απο τα ενεργά μέλη της κοινότητας αλλά και ολόκληρων κύκλων μαθημάτων με την πιστοποίηση της Unity, μπορεί να εξελιχθεί κάποιος απο το στάδιο του αρχάριου, στο στάδιο του επαγγελματία.

Community

Σε αυτό το κομμάτι, ένας χρήστης μπορεί να έρθει σε επικοινωνία με όλη την κοινότητα του Unity. Δηλαδή μπορεί να μπεί στο «Forum» και να διαβάσει άρθρα στις αντίστοιχες θεματικές εννότητες μέσα στο forum. Μπορεί να ρωτήσει, άλλους χρήστες και τους ίδιους τους developers, απορίες ή λύσεις σε ερωτήσεις άλλων μελών. Να δεί ανακοινώσεις και ενημερώσεις απο τους επίσημους αντιπρόσωπους της Unity και να πεί την γνώμη του πάνω σε αυτά. Η υποεννότητα «Answers» δίνει άμεσες λύσεις σε προβλήματα απο την Unity, μέσω της ενεργής του κοινότητας. Το «Feedback» χρησιμοποιείτε για να δώσει ιδέες, προβλήματα και λύσεις στην ίδια την εταιρία ωστε να βελτιωθεί προς το καλύτερο. Το «Issue Tracker» δίνει λύσεις σε κοινά και

αναγνωρισμένα προβλήματα που έχουν επιλυθεί, ώστε με λίγες κινήσεις να βρεί κανείς την απάντηση στο προβλημά του γρήγορα.

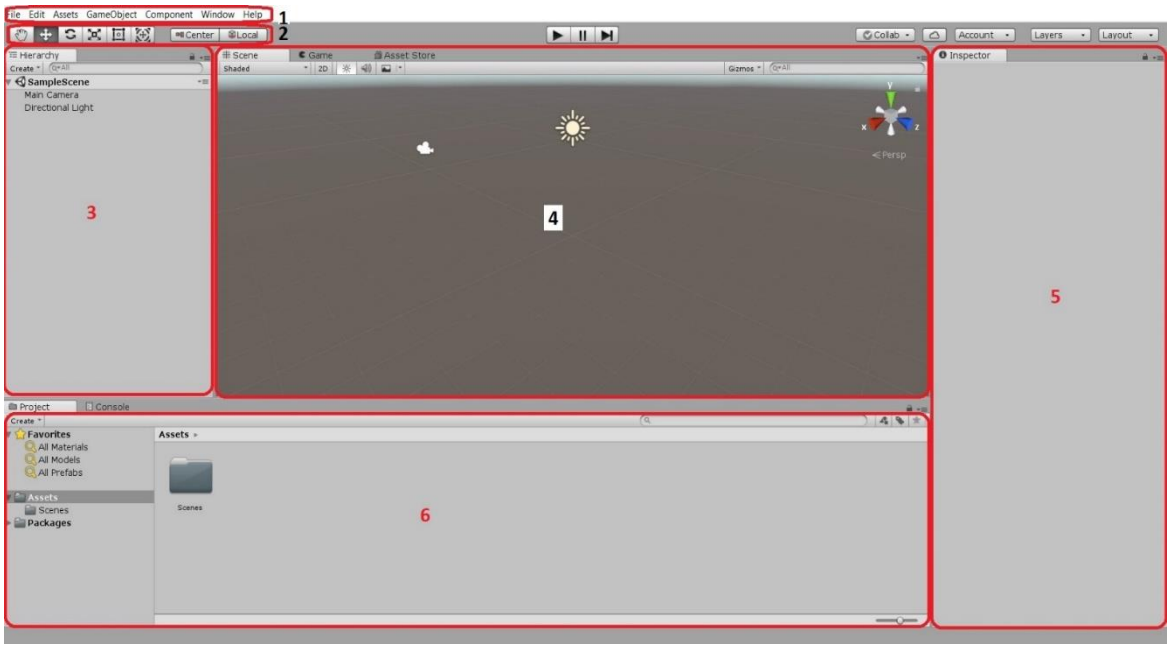
Get Unity

Ο τομέας αυτός αναφέρεται στην επιλογή πακέτου για την εγκατάσταση του λογισμικού. Βλέπουμε τα διαθέσιμα πλάνα με όλες τις λεπτομέριες και προϋποθέσεις. Στο τέλος της σελίδας υπάρχει ενα FAQ (Frequently Asked Questions = Συχνές ερωτήσεις) που αφορούν την συνδρομή. Αφού γίνει η επιλογή και η αποδοχή όρων, μπορούμε να κατεβάσουμε την εγκατάσταση. Οι απαραίτητες απαιτήσεις συστήματος είναι το λειτουργικό να είναι : Windows 7 SP1+, 8, 10, σε εκδόσεις 64-bit μόνο και macOS 10.11+. Και μια κάρτα γραφικών που υποστιρίζει DirectX10. Μας διευκρινίζει όμως, οτι από εκεί και πέρα οι απαιτήσεις αλλάζουν αναλόγως την περιπλοκότητα του project μας.

Asset Store

Ένα απο τα μεγαλύτερα πλεονεκτήματα της Unity είναι το Asset Store. Με αυτό, μας δίνει την ικανότητα να περιηγηθούμε σε μια τεράστια βιβλιοθήκη από δωρεάν ή επί πληρωμή asset που έχουν δημιουργηθεί απο την Unity Technologies είτε απο μέλη της κοινότητας. Τα assets καλύπτουν μεγάλη γκάμα όπως μοντέλα, υφή υλικού, κίνηση μοντέλων, επεκτάσεις του editor και πολλά άλλα. Οι χρήστες μπορούν να τα κατεβάσουν απο τον ίδιο τον editor απευθείας, αλλά και απο τον explorer τους. Τα assets επί πληρωμή μπορούν να χρησιμοποιηθούν και για προώθηση, αφού με την αγορά έχεις και τα πνευματικά δικαιώματα. Τα δωρεάν assets ανήκουν σε δύο κατηγορίες σε αυτά που έχουν περιορισμούς και αυτά που δεν έχουν περιορισμούς. Τα πρώτα χρησιμοποιούνται μόνο για εκπαιδευτικούς και προσωπικούς λόγους, ενώ τα δεύτερα μπορούν να χρησιμοποιηθούν επιπλέον και για εμπορικούς σκοπούς.

4.2 Η πρώτη επαφή με τον editor της Unity



Εικόνα 15 : Το κεντρικό παράθυρο της Unity

Αφού κατεβάσουμε και κάνουμε εγκατάσταση την Unity, σύμφωνα με τις οδηγίες, ανοίγοντάς την και δημιουργόντας ενα καινούργιο project, βλέπουμε την παρακάτω οθόνη. Να σημειωθεί ότι κάθε παράθυρο μπορει να μετακινηθεί και να διαμορφοθεί όπως επιθυμεί ο χρήστης.

Τα παρακάτω νούμερα αντιστοιχούν και επεξηγούν τα κόκκινα πεδία με τα αντίστοιχα νούμερα.

1) **Γραμμή Menu του editor**
2) **Μπάρα Εργαλείων του editor**
3) **Παράθυρο Ιεραρχίας (Hierarchy)**
4) **Κύριο παράθυρο**
5) **Παράθυρο Επιθεώρησης (Inspector)**
6) **Παράθυρο του project**

Αναλυτικότερα :

Η γραμμή menu του editor αποτελείτε απο διάφορα μέρη τα οποία μας επιταγχύνουν κάποιες διεργασίες. Συγκεκριμένα, αποτελείτε απο το «File», που μας επιτρέπει να διαχειριστούμε το project μας στην Unity. Δηλαδή να αρχίσουμε, να ανοίξουμε ένα project και να το σώσουμε. Μπορούμε επίσης να προσθέσουμε μια καινούργια «σκηνή» και να εξάγουμε το τελικό αποτέλεσμα του project μας, μέσω του μεταφραστή, στις πλατφόρμες που επιθυμούμε.

Ύστερα, το «Edit» μας επιτρέπει να χρησιμοποιήσουμε κάποιες βασικές επιλογές του editor. Όπως για παράδειγμα, να επεξεργαστούμε μια κίνηση που κάναμε μέσω του redo και undo, να τρέξουμε και να δοκιμάσουμε το project μας στο παράθυρο «Game», χωρίς να χρειάζεται να το εξάγουμε με τον μεταφραστή κάθε φορά που δοκιμάζουμε μια αλλαγή ή προσθέτουμε κάτι. Μπορούμε να επεξεργαστούμε ενα αντικείμενο ή αντικείμενα στην σκηνή μας με πολλούς τρόπους. Δηλαδή, να επιλέξουμε όλα τα αντικείμενα στην σκηνή, να φτίαξουμε μια μνήμη επιλογής συγκεκριμένων αντικειμένων που μπορούμε να ανακαλέσουμε ύστερα, να φτιάξουμε το διπλότυπο ενός αντικειμένου ή να διαγράψουμε τα επιλεγμένα αντικείμενα.

Το «Asset», όπως το προδίδει το ονομά του, ασχολείτε με την χρήση των asset. Δηλαδή την εισαγωγή ενός asset ή ενός πακέτου που εμπεριέχει πολλά asset, την δημιουργία ενός καινούργιου, την μετονομασία και την εξαγωγή του asset στην μορφή *.asset .

Το «Gameobject» αναφέρεται στην χρήση μιας μεγάλης βιβλιοθήκης αντικειμένων. Αυτά τα αντικείμενα μπορεί να είναι ένα γεωμετρικό σχήμα στον τρισδιάστατο ή δισδιάστατο χώρο, μία πηγή ήχου, μία πηγή φωτός, μία κάμερα. Όλα αυτα εμφανίζονται στο «Hierarchy» (βλέπε παρακάτω) και στο κύριο παράθυρο. Πλέον τα αντικείμενα αυτά θεωρούντε οντότητες στην σκηνή μας. Τέλος, μπορούμε να δημιουργήσουμε μια «κενή» οντότητα η οποία με τον ανάλογο κώδικα μπορεί να έχει μια συγκεκριμένη συμπεριφορά, που θα έχει αναθέσει ο προγραμματιστής.

Η καρτέλα «Components» είναι και αυτή μια βιβλιοθήκη ιδιοτήτων και μεταβλητών που χαρακτηρίζουν τα αντικείμενα στην σκηνή. Κάθε μια επιλογή προσφέρει συγκεκριμένα χαρακτηριστικά στον «Inspector» (βλέπε παρακατω) του αντικειμένου μαζί με πεδία που μπορούμε να αλλάξουμε τις μεταβλητές τροποποιώντας αυτή την ιδιότητα.

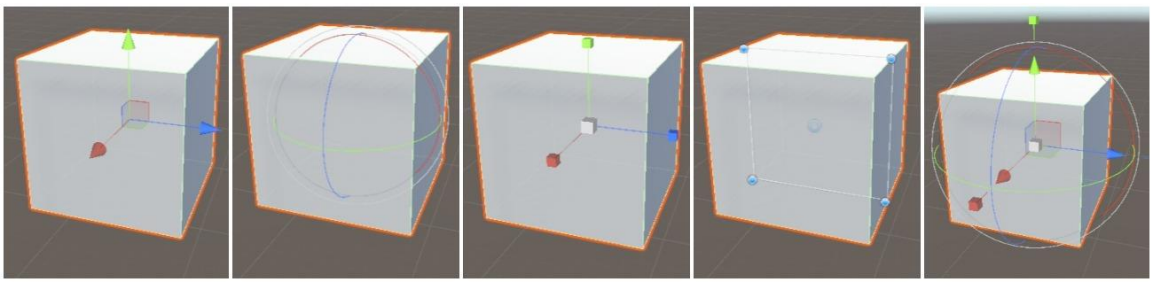
Στο «Window» μπορούμε να επιλέξουμε επιλογές σχετικά με τα παράθυρα του «Hierarchy», «Inspector», «Scene» και «Project». Είμαστε ικανοί να αλλάξουμε την διαρύθμισή τους, την εναλλαγή μεταξύ τους και την δημιουργία νέων παράθυρων, όπως το «Animator» για παράδειγμα που χρησιμοποιείτε για την αυτόματη κίνηση ενός αντικειμένου στην σκηνή μας.

Τέλος το «Help» αφορά την επικοινωνία μας με την Unity Technologies. Οι επιλογές αυτής της καρτέλας, μας ενημερώνει για την έκδοση λογισμικού Unity που χρησιμοποιούμε, τον έλεγχο καινούργιας και την επιλογή για εγκατάσταση της δοκιμαστικής έκδοσης. Μέσω άλλων επιλογών, μπορούμε να μεταβούμε στις αντίστοιχες ιστοσελίδες της κοινότητας της Unity με ευκολία.

Εικόνα 16 : Η μπάρα εργαλείων του editor

Όπως βλέπουμε στην παραπάνω εικόνα (Εικόνα 16), η μπάρα εργαλείων του editor αποτελείτε απο έξι κουμπιά. Το πρώτο (Hand tool) μπορούμε να σύρουμε την κάμερα,

κρατώντας το αριστερό κουμπί στο ποντίκι. Κρατώντας το Alt και το αριστερό κουμπί, μπορούμε να περιστρέψουμε την κάμερα γύρω απο τον εαυτό της. Τέλος κρατώντας το Alt και το δεξί πλήκτρο μπορούμε να ζουμάρουμε μέσα και έξω. Τα υπόλοιπα κουμπιά μας επιτρέπουν να χειριστούμε ένα αντικείμενο μέσα στην σκηνή, αφού το έχουμε επιλέξει. Τότε πάνω στο αντικείμενο εμφανίζονται βοηθητικές επιλογές, όπως βλέπουμε στην παρακάτω εικόνα και λειτουργούν ως εξής.



Εικόνα 17 : Τα διάφορα εργαλεία του editor πάνω στο αντικείμενο

Με το δεύτερο κουμπί μπορούμε να μετακινήσουμε ένα αντικείμενο (Move Tool) κρατώντας την μύτη ενός απο τα τρία βέλη και σέρνοντας το ποντίκι στην κατεύθυνση που θέλουμε και στις τρείς διαστάσεις. Αν όμως διαλέξουμε μία απο τις τρείς πλευρες, εκεί που εφάπτονται τα βέλη, τότε επιτρέπονται κινήσεις μόνο στις άλλες δύο διαστάσεις, δηλαδή εκτός από την πλευρά που έχουμε επιλέξει. Με το τρίτο κουμπί (Rotate Tool) μπορούμε να περιστρέψουμε το αντικείμενο πάλι προς όποια κατεύθυνση θέλουμε. Το τέταρτο κουμπί (Scale Tool) μας επιτρέπει να αλλάξουμε το μέγεθος του αντικειμένου. Με το άσπρο κουτι στο κέντρο το μεγαλώνουμε ή το μικραίνουμε κρατώντας το ίδιο σχήμα. Με τα άλλα τρία, το αλλάζουμε προς τον άξονα που επιλέξαμε. Το πέμπτο κουμπί (Rect Tool) μεγαλώνει το κουτί στην εικόνα ως προς την ακμή του κύβου. Τέλος, το τελευταίο κουμπί εμπεριέχει το Move Tool, το Rotate Tool και το Scale Tool ολα σε ένα για τον πλήρη έλεγχο του αντικειμένου με ένα κουμπί.

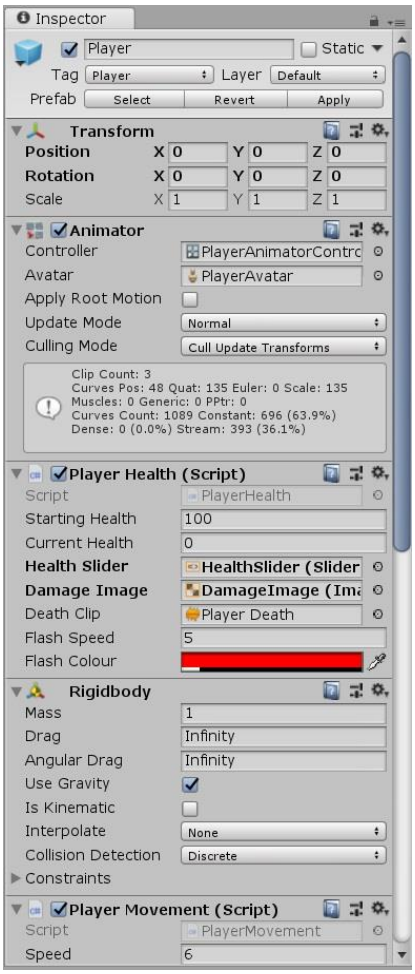
Εικόνα 18 : Παράθυρο Ιεραρχίας

Το παράθυρο Ιεραρχίας (Hierarchy), όπως βλέπουμε με το παράδειγμα της εικόνας, είναι μια λίστα που περιέχει τα αντικείμενα στην συγκεκριμένη σκηνή. Μπορεί να είναι assets ή προκατασκευασμένες οντότητες, σύνολο πολλών αντικειμένων με την ονομασία prefab. Μια επιλογή στο Hierarchy είναι η γονεϊκότητα. Τα αντικείμενα δηλαδή μπορεί να έχουν σχέση γονέα-παιδιού και να μοιράζονται χαρακτηριστικά. Παράδειγμα γονέα-παιδιού φαίνεται στην εικόνα. Το «Player» είναι ο γονέας και λειτουργεί σαν φάκελος για τα επόμενα αντικείμενα, το «Player», το «Gun» και το «GunBarrelEnd» το οποίο είναι γονέας του «Facelight», κάνοντας το «Facelight» παιδί του «GunBarrelEnd» αλλά και του «Player».

Το κύριο παράθυρο λειτουργεί για την γραφική αναπαράσταση όλου του project. Στην καρτέλα «Scene» μπορούμε να διαχειριστούμε τα GameObjects, φτιάχνωντας τα επίπεδα, στήνωντας τις κάμερες που βλέπει ο χρήστης, στήνωντας το φώς του επιπέδου. Γενικά, είναι το παράθυρο που σκηνοθετεί ο Game Designer, πως θα δείχνει το τελικό προϊόν στον παίχτη. Η άλλη καρτέλα «Game» λειτουργεί για να μας δείξει το τελικό προϊόν, χωρίς να το κάνουμε compile μέσω του μεταφραστή. Άρα μπορούμε να δοκιμάζουμε αλλαγές ή να βρούμε bugs(=προβλήματα στον κώδικά μας) γρήγορα. Οι αλλαγές μπορεί να δοκιμαστούν και κατα την διάρκεια του «Play», διότι αποθηκεύει την κατάσταση πριν το «Play» και οταν πατήσουμε «Stop» μας επαναφέρει στις προηγούμενες ρυθμίσεις. Το μειονέκτημα σε αυτήν την τεχνική είναι οτι πρέπει να σημειώσουμε ή να θυμόμαστε τις αλλαγές διότι μετά το «Stop», χάνονται όλες οι ρυθμίσεις.

Εικόνα 19 : Κουμπιά έναρξη και παύσης σκηνής

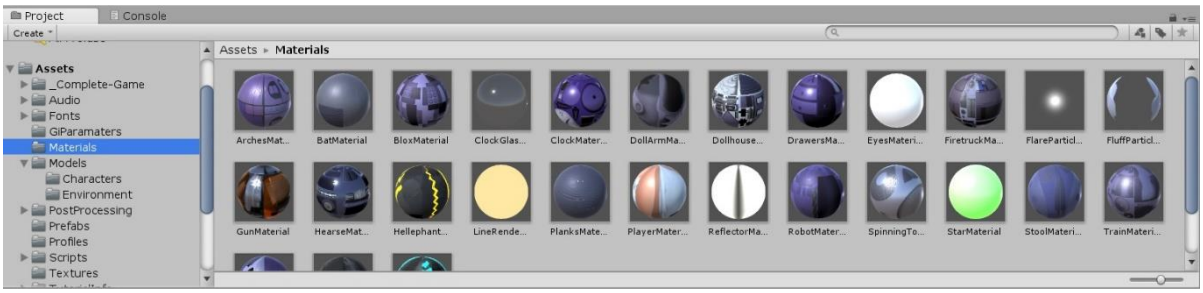
Κάθε αντικείμενο που επιλέγουμε στο κύριο παράθυρο έχει κάποιες ιδιότητες είτε μπορούμε να προσθέσουμε κάποιες. Το παράθυρο επιθεώρησης (Inspector) βοηθάει σε αυτό. Δηλαδή μπορούμε να προσθέσουμε μεταβλητές, από την καρτέλα «Components» (βλέπε Γραμμή menu), στα αντικείμενα για να τα ελέγξουμε. Τα components είναι script στην βιβλιοθήκη της Unity με μεταβλητές. Καθ' άυτο τον τρόπο μπορούμε να δημιουργήσουμε δικά μας script και να εμφανίζουμε στον Inspector ότι θέλουμε να μεταβάλουμε, χωρίς να χρειάζεται αλλαγή απο το ίδιο το script. Όπως βλέπουμε και στην εικόνα, κάθε μοναδικό χαρακτηριστικό είναι μια εννότητα στον Inspector και έχει τις δικές του μεταβλητές.



Εικόνα 20 : Παράθυρο Επιθεώρησης

Με το παράθυρο του project (Project Window), έχουμε πρόσβαση στα asset μας. Κάθε φορά που κάνουμε import (= Εισαγωγή στην Unity) ένα asset εμφανίζεται και σε αυτό το παράθυρο. Δεν είναι αναγκαίο να το χρησιμοποιούμε και στην σκηνή μας, οπότε λειτουργεί και σαν «βιβλιοθήκη». Εάν θέλουμε να το βάλουμε στην σκηνή, ενα drag and drop αρκεί. Αυτό με την σειρά του θα εμφανιστεί στο Hierarchy ώστε να το τροποποιήσουμε όπως θέλουμε. Ισχύει όμως και το ανάποδο. Εάν έχουμε δημιουργήσει ένα object μέσα στην σκηνή και θέλουμε να το χρησιμοποιήσουμε ξανά ή να το έχουμε

διαθέσιμο για άλλα project, μπορούμε να δημιουργήσουμε ένα prefab με τα χαρακτηριστικά του object την στιγμή που το δημιουργήσαμε.



Εικόνα 21 : Παράθυρο Project και τα διάφορα asset στον φάκελο Materials

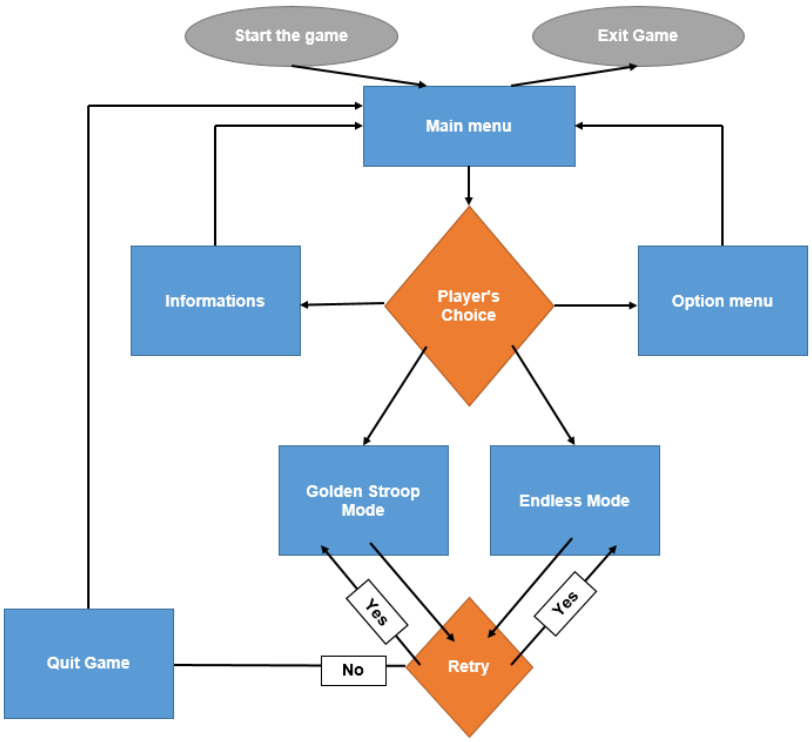
5 Η Εφαρμογή

5.1 Εισαγωγή

Για την εφαρμογή του Stroop Effect αποφασίσαμε να δημιουργήσουμε μια εφαρμογή 2D puzzle game με την βοήθεια της Unity. Στόχος της εφαρμογής ήταν να υποβάλουμε τον χρήστη στο Stroop τέστ με έναν σύγχρονο τρόπο και παιχνιδοποιώντας το τρόπο διεξαγωγής του τεστ. Η έκδοση που βασίστηκε το παιχνίδι ήταν η έκδοση του Charles J. Golden, περίπου το 1978. Η διαφορά βέβαια με το ίδιο το τεστ είναι οτι τα παραδείγματα λέξεων και χρωμάτων είναι όλα τυχαία, για να λειτουργήσει σαν ένα παιχνίδι χωρίς ένα επαναλαμβανόμενο μοτίβο. Επειδή τα assets χρειάζονται γνώσεις απο επεξεργασία ήχου, επεξεργασία εικόνας σε 2D ή 3D, κάποια assets δημιουργήθηκαν, επεξεργαστήκαν και χρησιμοποιήθηκαν απο εμάς, ενώ κάποια άλλα χρησιμοποιώντας το Unity Asset Store τα χρησιμοποιήσαμε στο παιχνίδι. Η λίστα των asset που χρησιμοποιήθηκαν βρίσκεται στο τέλος. Να σημειωθεί οτι το παιχνίδι είναι στα Αγγλικά.

5.2 Σχεδιασμός

Σύμφωνα με το παρακάτω διάγραμμα ροής, μπορούμε να δούμε εικονικά όλες τις επιλογές που μπορεί να κάνει ο χρήστης με το πρόγραμμα. Παρακάτω αναλύουμε αναλυτικότερα κάθε επιλογή.



Εικόνα 22 : Διάγραμμα ροής του συστήματος

Αρχικό μενού

Για το **αρχικό μενού**, έχουμε κάποια βασικά κουμπία, που αφορούν την βασική λειτουργία του προγράμματος, και κάποια δευτερεύοντα, για πληροφορίες. Μπορούμε να δούμε το αρχικό μενού στην παρακάτω φωτογραφία

Εικόνα 23 : Το αρχικό μενού της εφαρμογής

Βασικά κουμπία:

[Golden Stroop / Endless Mode] Αυτά τα δύο κουμπία μας οδηγούν στην οθόνη του παιχνιδιού, ή αλλιώς «Σκηνή». Είναι δύο διαφορετικές λειτουργίες / τρόποι παιχνιδιού με άλλους κανόνες.

[Options] Το κουμπί μας επιτρέπει να δούμε τις ρυθμίσεις, όπως για παράδειγμα την ενεργοποίηση / απενεργοποιήση του ήχου και την διαγραφή των αποθηκευμένων High score.

[Exit] Το κουμπί μας επιτρέπει να κλείσουμε την εφαρμογή.

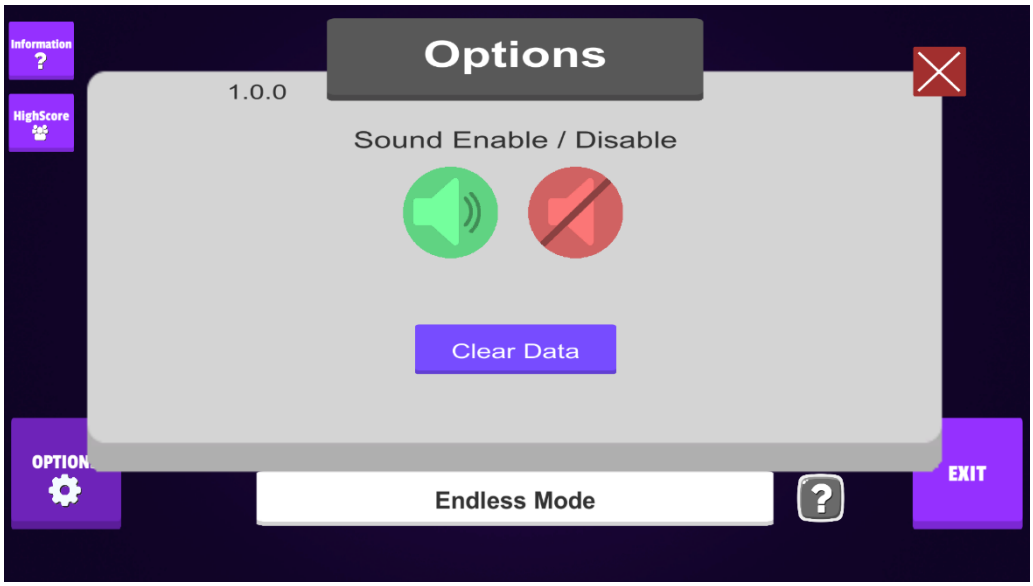
Δευτερεύοντα κουμπία:

[Information] Σε αυτό το κουμπί μας δίνονται κάποιες βασικές πληροφορίες για τον Stroop και το πείραμά του, για να μπορεί ο χρήστης να κατανοήσει το παιχνίδι.

[High Score] Το κουμπί μας μεταφέρει σε ένα καινούργιο παράθυρο και μας δείχνει το καλύτερη βαθμολογία που έχουμε καταφέρει στις δύο λειτουργίες της εφαρμογής.

[?] Μας εμφανίζουν μια σύντομη επεξήγηση των κανόνων για κάθε μία απο τις λειτουργίες της εφαρμογής, με ενα αναδυόμενο παράθυρο.

Οθόνη Ρυθμίσεων

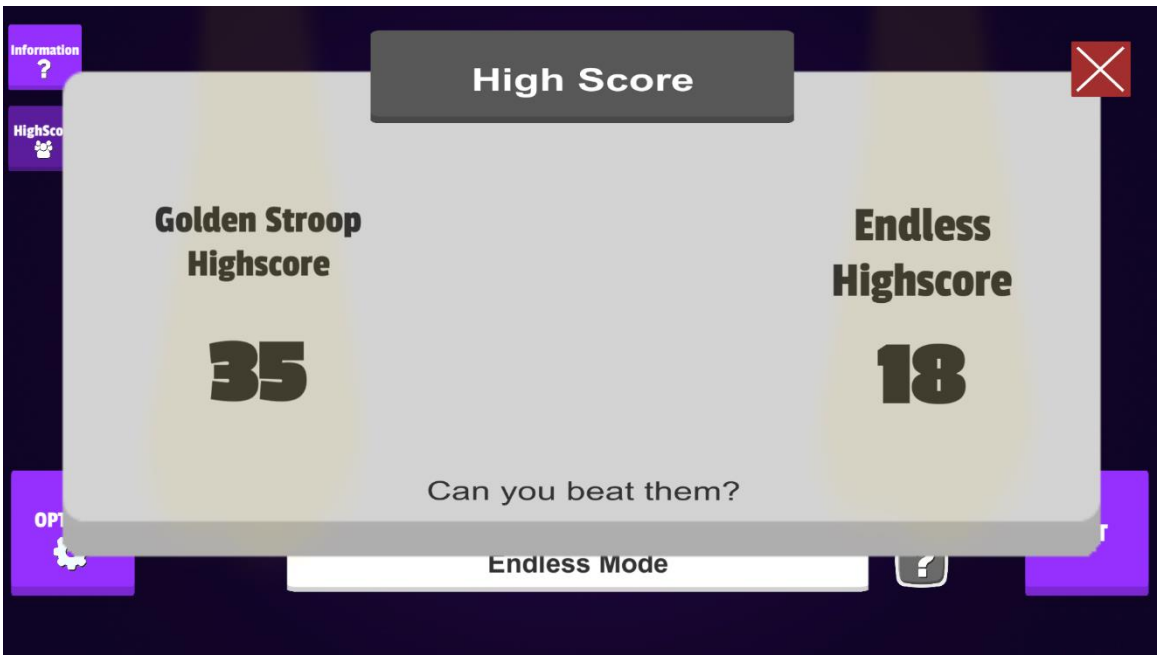


Εικόνα 24 : Οθόνη ρυθμισεων της εφαρμογής

Όπως βλέπουμε στην παραπάνω εικόνα, έχουμε δύο κουμπία ενεργοποιήσης / απενεργοποιήσης ήχου. Πάνω αριστερά ο αριθμός «1.0.0» δηλώνει την έκδοση της εφαρμογής. Τέλος το κουμπί «Clear Data» διαγράφη απο την μνήμη του υπολογιστή την βαθμολογία του.

Οθόνη Βαθμολογίας

Η οθόνη βαθμολογίας μας δείχνει την υψηλότερη αποθηκευμένη βαθμολογία για τις δύο διαφορετικές λειτουργίες της εφαρμογής. Να σημειωθεί οτι οι αριθμοί 35 και 18 είναι παραδείγματα λειτουργίας του συστήματος.



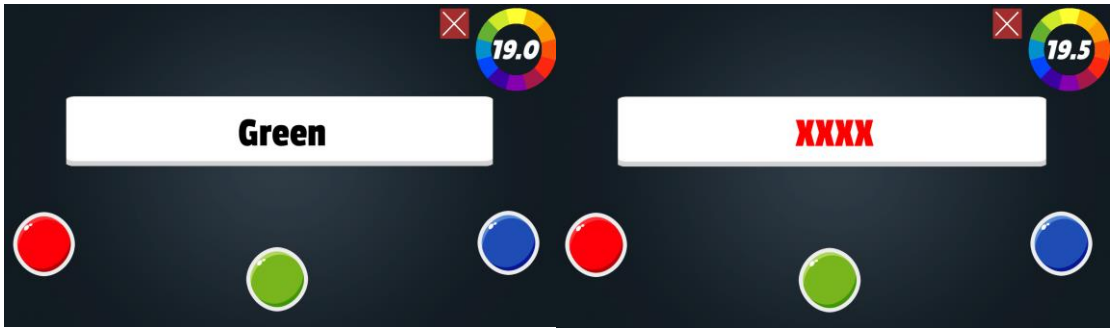
Εικόνα 25 : Οθόνη Βαθμολογίας της εφαρμογής

Λειτουργία Golden Stroop

Είναι η βασική λειτουργία του παιχνιδιού. Πάνω δεξιά οι αριθμοί μας δείχνουν χρόνο που μας απομένει. Δίπλα, το κουμπί [X] είναι το κουμπί εξόδου, που μας επιστρέφει στο αρχικό μενού. Στην μέση εμφανίζεται ένα άσπρο πλαίσιο που μικραίνει συνέχεια μέχρι να εξαφανιστεί, λειτουργώντας σαν γραφική απεικόνιση του χρόνου. Πάνω στο πλάισιο εμφανίζεται το τεστ, που θα αναλύσουμε παρακάτω. Τέλος, υπάρχουν τρία κουμπιά σε χρώμα κόκκινο, πράσινο και μπλέ, που εξυπηρετούν τον χρήστη να απαντήσει στο τεστ.

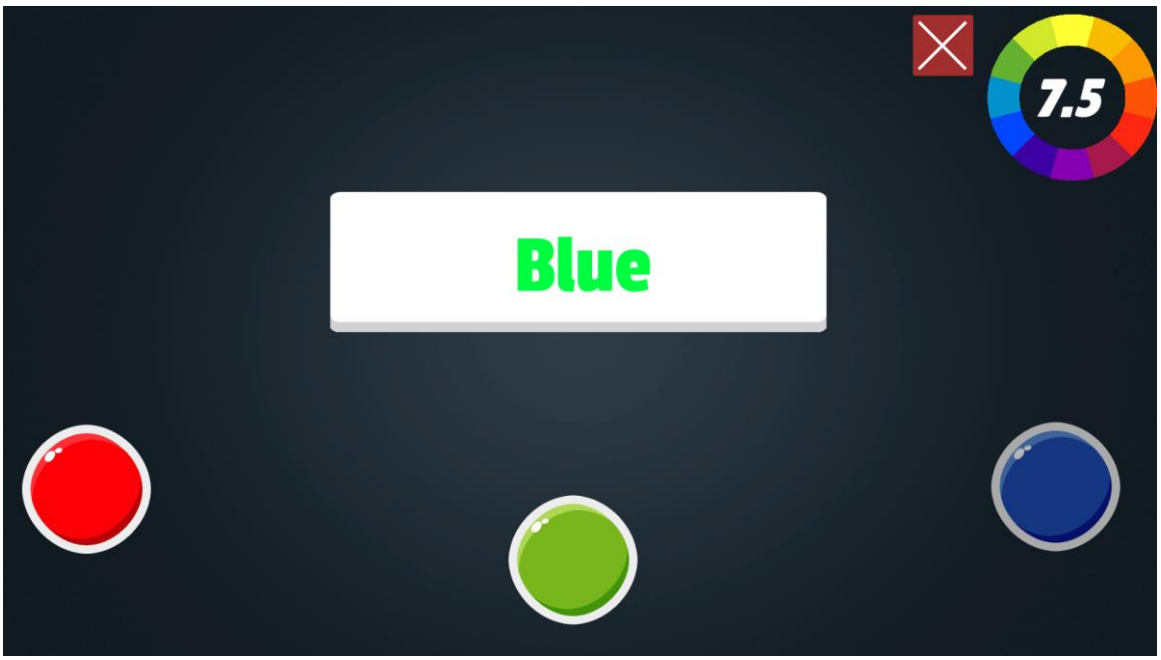
Το παιχνίδι αποτελείτε απο τρείς γύρους. Κάθε ενας διαρκεί έως και είκοσι (20) δευτερόλεπτα και εμφανίζονται μέχρι και σαράντα (40) λέξεις-χρώματα. Κάθε σωστή απάντηση μας δίνει ένα (1) πόντο και δεν υπάρχει επιπτώσεις στην λάθος απάντηση. Σύνολο το τέστ διαρκεί ένα (1) λεπτό και ο χρήστης μπορεί να φτάσει τον μέγιστο αριθμό των εκατόν είκοσι (120) πόντων.

Στον πρώτο γύρο το παιχνίδι παρουσιάζει με τυχαία σειρά τις λέξεις «Red», «Green», «Blue» και ο παίχτης πρέπει να διαλέξει το αντίστοιχο χρώμα απο τα κουμπία στο κάτω μέρος τις οθόνης. Στον δεύτερο γύρο, ο παίχτης καλείτε να απαντήσει σωστά πάλι σύμφωνα με το χρώμα, όμως τώρα εμφανίζονται η λέξη «XXXX» με χρώμα γραματοσειράς κόκκινη, πράσινη και μπλέ.



Εικόνα 26 Αριστερά ο πρώτος γύρος και δεξιά ο δεύτερος γύρος της λειτουργίας Golden Stroop

Στον τρίτο και τελευταίο γύρο, εμφανίζονται τυχαία οι λέξεις «Red», «Green», «Blue» και επιπλέον έχουν τυχαία γραμματοσειρά χρώματος κόκκινη, πράσινη και μπλέ. Ο παίχτης για να θεωρηθεί η απάντησή του σωστή πρέπει να πατήσει το σωστό κουμπί σύμφωνα με την λέξη και όχι με το χρώμα γραμματοσειράς της λέξης. Παρακάτω βλέπουμε ενα παράδειγμα, στο οποίο η σωστή απάντηση είναι το Μπλε!



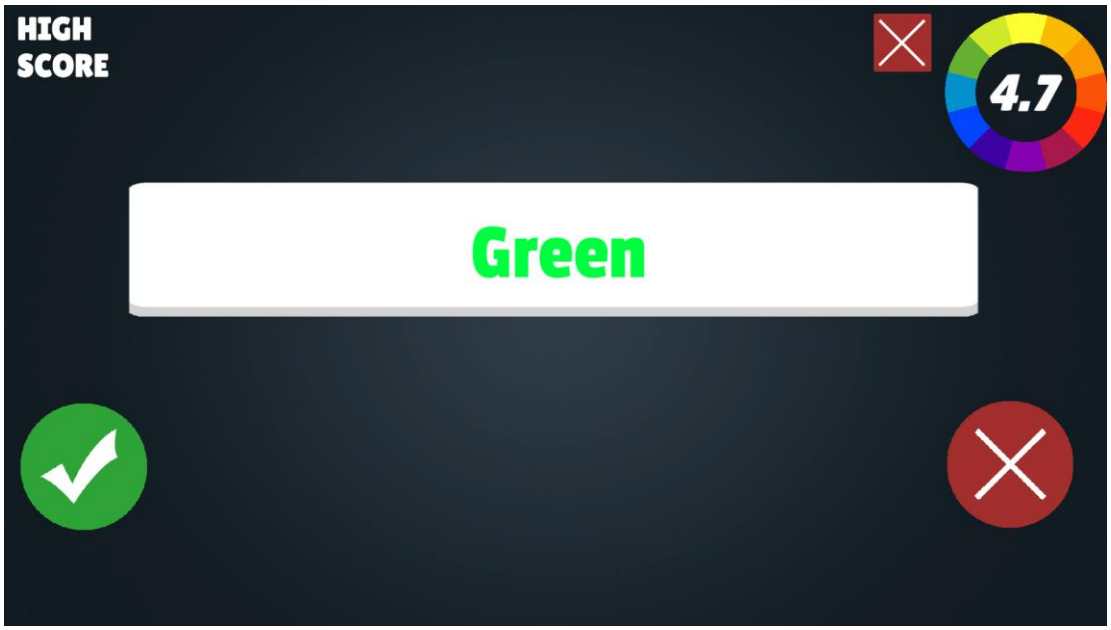
Εικόνα 27 Τρίτος γύρος της λειτουργίας Golden Stroop

Λειτουργία Endless Mode

Ονομάζεται Endless, ή στα ελληνικά χωρίς τέλος, διότι οι λέξεις-χρώματα δεν σταματάνε να εμφανίζονται, μέχρι ότου ο παίχτης κάνει λάθος ή ο χρόνος μηδενίσει. Σε αυτή την λειτουργία έχουμε πάνω αριστερά μια υπενθύμιση του προηγούμενου High Score, το οποίο προσπαθούμε να ξεπεράσουμε. Επίσης τώρα έχουμε μόνο δύο κουμπιά. Το σωστό, πράσινο κουμπί, και το λάθος, κόκκινο κουμπί.

Εάν η λέξη-χρώμα που εμφανίζεται, έχει ίδιο χρώμα γραμματοσειράς τότε ο παίχτης πρέπει να πατήσει το πράσινο «Σωστό» κουμπί. Σε οποιαδήποτε άλλη

περίπτωση, ο παίχτης πρέπει να πατήσει το κόκκινο «Λάθος» κουμπί, ώστε να απαντήσει σωστά. Στο παράδειγμα της εικόνας ο χρήστης καλείται να πατήσει το πράσινο «Σωστό» κουμπί μιάς και το «Green» αντιστοιχεί στο πράσινο χρώμα της γραμματοσειράς.



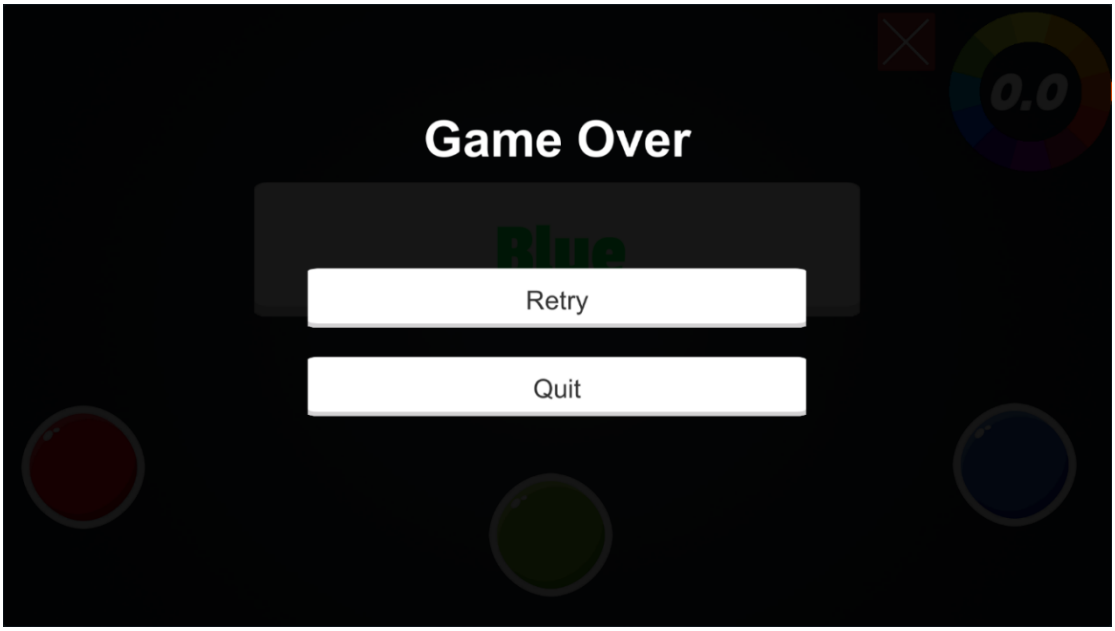
Εικόνα 28 : Η λειτουργία Endless

Οθόνη Game Over

Εμφανίζεται όταν τελειώνει ενα παιχνίδι. Οι επιλογές του χρήστη είναι μόνο δύο.

[Retry] Πατώντας το κουμπί ο χρήστης κάνει επανεκκίνηση της λειτουργίας, προσπαθώντας πάλι απο την αρχή.

[Quit] Πατώντας το κουμπί ο χρήστης μεταφέρεται στο αρχικό μενού.



Εικόνα 29 : Η οθόνη Game Over

Χειρισμός

Ο χρήστης μπορεί να περιηγηθεί σε όλο το πρόγραμμα με την χρήση του ποντικιού. Προσθέσαμε όμως και χειρισμό του προγράμματος με το πληκτρολόγιο. Αυτό βελτιώνει την απόδοση στο τέστ, διότι ο χρήστης δεν χάνει χρόνο μεταφέροντας το ποντίκι ανάμεσα στα κουμπιά ελέγχου απάντησης. Γι'αυτό και ο χειρισμός πληκτρολογίου είναι απλός.

 Αριστερό/Δεξί/Κάτω Βελάκι : Επιλέγει το κουμπί απάντησης του τέστ.

 Space Bar : Λειτουργεί σαν «OK» και για επιλογή του «Retry».

 Escape : Λειτουργεί σαν «Πίσω» ή έξοδος του παιχνιδιού.

5.3 Μελλοντικές Αναβαθμίσεις

Η εφαρμογή έχει πολλές προοπτικές για αναβάθμιση. Η μεταφορά του σε άλλες πλατφόρμες όπως Android και iOS είναι πολύ πιθανή και θα είναι πιο ευχάριστη μιάς και που τα κουμπιά θα ενεργοποιούνται με τα δάκτυλα των χρηστών σε πιο μικρή οθόνη. Αυτό μπορεί να αυξήση και την αποδοτικότητα στο τέστ.

Σύμφωνα με το κεφάλαιο 3.3, οι δυνατότητες αναβαθμίσεων ποικίλουν διότι υπάρχουν πολλές τροποποιήσεις του αρχικού τεστ του Stroop. Μιά άμεση αναβάθμιση που θα παρακινήσει τους χρήστες της εφαρμογής να κάνουν καλύτερη βαθμολογία, είναι φτιάχνοντας ένα πίνακα κατάταξης, είτε παγκόσμιο είτε σχέση με τους φίλους τους.

Ακόμα θα μπορούσαμε να έχουμε μια επιλογή γλώσσας στο παράθυρο των ρυθμίσεων με κάποιες βασικές γλώσσες, γιατί τα αγγλικά μπορεί να μην είναι κατανοητά απο όλους ή να έχουμε αλλαγή στην ταχύτητα αναγνώρισης της λέξεως στην μητρική γλώσσα προς το καλύτερο ή προς το χειρότερο.

6 Υλοποίηση Παιχνιδιού

6.1 Εισαγωγή

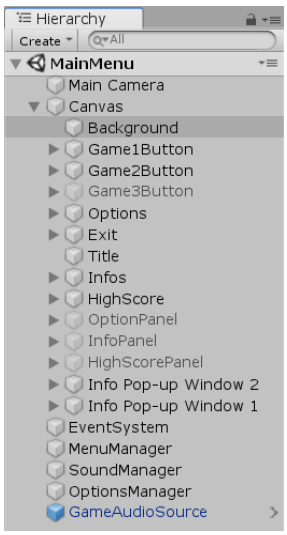
Σαν βάση βρήκαμε ένα asset το οποίο και τροποποιήσαμε για τα επιθυμητά αποτελέσματα. Ο κώδικας που χρησιμοποιείται είναι η C#, που δημιουργήθηκε απο την Microsoft για το .Net Framework. Για την τροποποίηση του κώδικα χρησιμοποιήθηκε το Visual Studio 2019. Για την επεξεργασία εικόνων χρησιμοποιήθηκε το λογισμικό Gimp, ενώ για την επεξεργασία ήχων το Audacity.

Σε αυτό το κεφάλαιο θα γίνει περιγραφή του κώδικα και θα χρησιμοποιηθούν κομμάτια κώδικα, κυρίως απο την λειτουργία «Golden Stroop».

6.2 Αρχικό μενού

Για το αρχικό μενού χρησιμοποιούμε την σκηνή «MainMenu» στην Unity. Φτιάξαμε την σκηνή βάζοντας στην Ιεραρχία Game Objects, τα οποία φαίνονται στην εικόνα. Τα Game Objects που βρίσκονται στην κατηγορία Canvas, όλα λειτουργούν οπτικά στον χρήστη ως UI, ενώ τα υπόλοιπα έχουν συνημμένα C# script, που βοηθούν στην συμπεριφορά και την λογική του αρχικού μενού.

Παρακάτω αναθέτουμε και τον κώδικα του OptionManager.cs που ελέγχει την συμπεριφορά του αρχικού μενού.

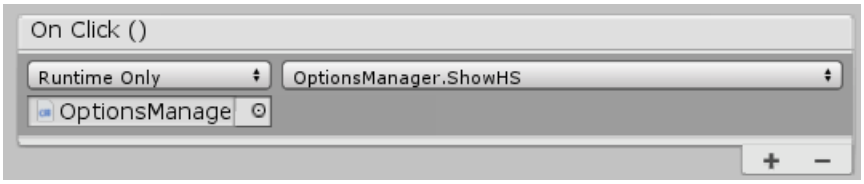


Εικόνα 30 : Τα game objects του MainMenu όπως φαίνονται στην Ιεραρχία

```
public class OptionsManager : MonoBehaviour {


    public GameObject m_OptionsMenu; //The options menu canvas object
    public GameObject m_Info; //The info canvas object
    public GameObject m_Highscore; //The info canvas object
    public GameObject m_Lights; //The Lights object
    public Text m_VersionNumberText; //Our version number to display
    private int highScore; //Our highscore for endless mode from local files
    private int GhighScore; //Our highscore for Golden Stroop mode from local files
    public Text m_HighScoreText; //The variable that saves Highscore for Endless Mode
    public Text m_GHighScoreText; //The variable that saves Highscore for Golden
Stroop Mode
    /// <summary>
```

Κώδικας 1 Option Manager

Οι λειτουργίες των κουμπιών γίνονται απο τα UI Buttons στο Canvas. Κάθε UI Button έχει διαφορετική συμπεριφορά και εικόνα, μιάς και που κάθε ένα εξυπηρετεί διαφορετικό σκοπό. Ο OptionManager έχει συναρτήσεις και μεθόδους που καλούνται όταν ο χρήστης πιέσει ένα κουμπί. Παραδείγματος χάρη, για να ανοίξει το παράθυρο «High Score», καλείται απο το κουμπί «HighScore» η μέθοδος ShowHS() μέσω του Inspector της Unity απο την παρακάτω εικόνα.



Εικόνα 31 Επιλογή OnClick() μέσω του Editor

Ύστερα εκτελείτε απο το OptionManager.cs η παρακάτω μέθοδος. Για να κλείσει χρησιμοποιείται το κουμπί [X] πάνω δεξιά και καλείται η μέθοδος CloseHS().

```
    public void ShowHS()
    {
        m_OptionsMenu.SetActive(false); //Closes Option menu if it is open
        m_Info.SetActive(false); //Closes information if it is open
        highScore = PlayerPrefs.GetInt("HighScore1"); //Gets the highscore for Endless Mode from local
files
        GhighScore = PlayerPrefs.GetInt("GoldenHighScore"); //Gets the highscore for Golden Stroop
from local files
        m_HighScoreText.text = "" + highScore; //Changes the text of UI.Button.text to show highscore
of Endless mode
        m_GHighScoreText.text = "" + GhighScore; //Changes the text of UI.Button.text to show
highscore of Golden Stroop mode
        m_Highscore.SetActive(true); //Set highscore screen ON
        m_Lights.SetActive(true); //Set Lights ON
    }
```

Κώδικας 2 Ο κώδικας για την ενεργοποίηση / απενεργοποίηση του High score window

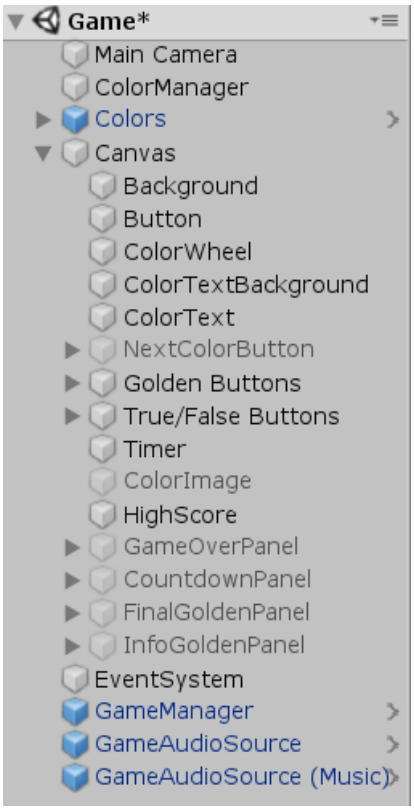
Η επιλογή των δύο διαφορετικών Game Modes γίνεται απο το MenuManager.cs . Όταν πατηθεί ένα απο τα δύο κουμπιά καλείται η LoadGame(), η οποία φορτώνει και την επόμενη σκηνή «Game».

```
public class MenuManager : MonoBehaviour {


        /// <summary>
        /// Loads the game.
        /// Give a game type number to load the correct linked game scene
        /// </summary>
        public void LoadGame(int gameType) {
                switch (gameType) {
                case 0:
                        GlobalVariables.GameToPlay = 0; ///Endless Mode


                        break;
                ///case 1:
                ///     GlobalVariables.GameToPlay = 1; Reserved For Future Game Mode
                ///     break;
                case 2:
                        GlobalVariables.GameToPlay = 2; ///Golden Stroop
                        break;
                default:
                    break;
                }


                SceneManager.LoadScene (GlobalVariables.GameScene); //Loads the game scene given
        }
}
```

Κώδικας 3 Οι περιπτώσεις για αλλαγή της λειτουργίας

6.3 Σκηνή «Game»

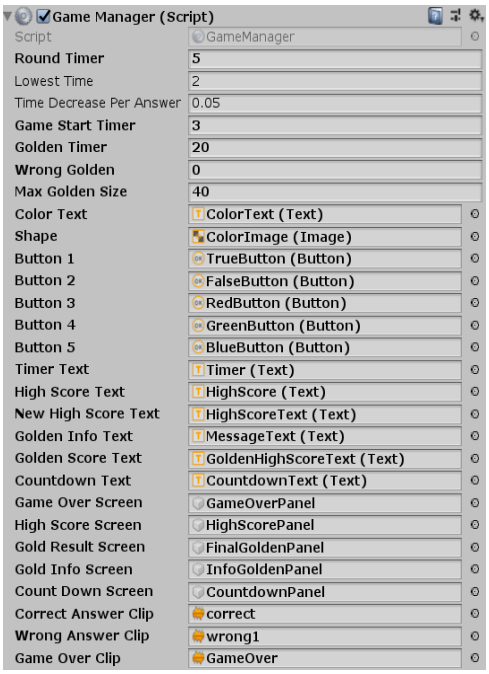
Η σκηνή «Game» χρησιμοποιείται και για τα δύο Game Modes. Για να επιτευχθεί αυτό χρησιμοποιούνται Game Objects του Canvas, τα οποία ενεργοποιούνται και απενεργοποιούνται αναλόγως την επιλογή του χρήστη. Η διπλανή εικόνα είναι η Ιεραρχία της σκηνής και βλέπουμε οτι τα Game Objects με γκρι χρώμα, είναι ανενεργά.



Εικόνα 33 : Τα αντικείμενα της σκηνής Game στην Ιεραρχία

Για να καθορίσουμε τα χρώματα, έχουμε μια κλάση ColorObject.cs η οποία έχει τα χαρακτηριστικά του ονόματος του χρώματος, καθώς και το χρώμα καθαυτό, σε 32-bit μορφή. Ύστερα ο ColorManager.cs διαχειρίζει τα χρώματα που θέλουμε συγκεντρωτικά. Πιο συγκεκριμένα, εμείς θέλουμε τρία χρώματα, με τα χαρακτηριστικά της κλάσης ColorObject.cs (όνομα και χρώμα σε 32-bit μορφή), το κόκκινο, το μπλέ και το πράσινο.

Όλη η λογική του παιχνιδιού βρίσκεται στο GameManager.cs . Στην εικόνα(Εικόνα 33) βλέπουμε τον Game Manager στον Inspector. Εδώ κάνουμε όλες τις αναφορές που διαχειρίζεται το script GameManager.cs . Για να εμφανίσουμε τις αναφορές στον Inspector χρειάζεται να τις αρχικοποιήσουμε ώς public στο C# script. Συγκεκριμένα έχουμε αναφορές σε μεταβλητές, εικόνες και κουμπιά του UI, κείμενα, Game objects και τέλος ήχους. Αυτό μας διευκολύνει ειδικά στις μεταβλητές γιατί μπορούμε άμεσα να αλλάξουμε μια τιμή για δοκιμές, χωρίς να χρειάζεται να επεξεργαζόμαστε το script.



Εικόνα 32 :Ο Game Manager στον Inspector

Ο Game manager μας βοηθάει να υπολογίζουμε τον χρόνο μέσω της FixedUpdate(). Ανήκει στις βιβλιοθήκες της Unity και είναι μια μέθοδος που έχουμε καλέσει, και τρέχει 50 φορές το δευτερόλεπτο. Επειδή καλείτε τόσο συχνά, χρησιμοποείτε αποκλειστικά για μέτρηση χρόνου, για να μην κάνει το πρόγραμμα «βαρύ». Επίσης δεν επηρεάζεται απο καθυστερήσεις του συστήματος που τρέχει το πρόγραμμα.

```
void FixedUpdate() {

        if (gameType == GameType.ColorWord)
        if (gameType == GameType.ColorShape)
        if (gameType == GameType.GoldenStroop)
        {
            //Check to see if the game has just started so that we can run the initial countdown until start
            if (gameStartCountdown)
            {
                m_GameStartTimer -= Time.deltaTime; //Start decreasing time towards 0
                m_CountdownText.text = "" + m_GameStartTimer.ToString("F1");
                if (m_GameStartTimer <= 0.0f)
                {
                    m_CountDownScreen.SetActive(false); //Disable the countdown timer screen
                    gameStartCountdown = false; //The countdown has ended, so toggle this off
                    isPaused = false; //The game will start now
                    spinner.StartRotation();
                    this.NewGolden();
                }
            }
            //Check to see if the game is paused right now, if not, do the countdown
            if (!isPaused)
            {
                m_GoldenTimer -= Time.deltaTime; //Delta time will help prevent time loss or skipping
                m_TimerText.text = "" + m_GoldenTimer.ToString("F1");
                //Check if the sample was given to Player

                if (GoldenSize >= maxGoldenSize)
                {
                    isPaused = true;
                    this.NextGoldenStage();
                }
                //Check for a out of time count on the answer
                if (m_GoldenTimer <= 0.0f)
                {
                    isPaused = true; //Pause the game as it is now over
                    m_TimerText.text = "0.0"; //Set the time to lock at 0
                    //Show results and enter next stage
                    this.NextGoldenStage();
                }
            }
        }
    }
```

Κώδικας 4 Η FixedUpdate() μέθοδος

Στον παραπάνω κώδικα φαίνονται δύο μέθοδοι βασικοί για το παιχνίδι. Ο πρώτος είναι ο NewGolden() και NextGoldenStage().

NewGolden()

Χρησιμοποιείται για να μας δώσει τον επόμενο συνδιασμό χρώματος, δηλαδή όνομα και χρώμα γραμματοσειράς. Αναλόγως σε ποιό στάδιο βρισκόμαστε επιλέγει εναν τυχαίο συνδιασμό και μας τον εμφανίζει. Επίσης υπάρχει μια μεταβλητή «GoldenColor», η οποία βοηθάει τον έλεγχο της απάντησης του χρήστη στην CheckAnser() ως σημαία.

```
    public void NewGolden()
    {
        //We need a color and word and don't want them to be the same, so we get two random references
        //in the array which will be used to determine the question and answer
        int randomWord = Random.Range(0, colorManager.m_GameColors.Length);
        int randomColor = Random.Range(0, colorManager.m_GameColors.Length);
        if (GoldenStage == 1)
        {
            m_ColorText.text = colorManager.m_GameColors[randomWord].m_ColorName;  //Gets a random color's
name and shows on screen
            m_ColorText.color = Color.black; //Sets the font's color to black
            GoldenColor = randomWord; //Flag for Check answer

        }
        else if (GoldenStage == 2)
        {
            m_ColorText.text = "XXXX"; //The text that appears instead of color's name
            m_ColorText.color = colorManager.m_GameColors[randomColor].m_Color; //Gets a random font color
and shows on screen along with text
            GoldenColor = randomColor; //Flag for Check answer
        }
        else
        {
            m_ColorText.text = colorManager.m_GameColors[randomWord].m_ColorName; //Gets a random color's
name and shows on screen
            m_ColorText.color = colorManager.m_GameColors[randomColor].m_Color; //Gets a random font color
and shows on screen along color's name
            GoldenColor = randomWord; //Flag for Check answer
        }
    }
```

Κώδικας 5 : Η μέθοδος NewGolden()

NextGoldenStage()

Είναι η μέθοδος που μας επιτρέπει να αλλάξουμε το στάδιο του Golden Stroop τέστ. Καλείτε όταν τελειώσουν οι λέξεις (το όριο λέξεων έχει οριστεί στις σαράντα) ή όταν τελειώσει ο χρόνος (το χρονικό περιθώριο είναι είκοσι δευτερόλεπτα). Ελέγχει αρχικά σε ποιό στάδιο είμαστε και εμφανίζει το παράθυρο με τις οδηγίες για το επόμενο στάδιο. Δουλεύει συνδιαστικά με την μέθοδο CloseGoldenResultScreen(), η οποία καλείται όταν ο χρήστης κλείσει το παράθυρο που εμφανίζει η NextGoldenStage(). Παρακάτω παραθέτουμε τον κώδικα και για τις δύο μεθόδους.

```
    public void NextGoldenStage()
    {
        //Checks which stage we are on and make transition between stages.
        if (GoldenStage == 1 )
        {
            GoldenStage=2;
            m_GoldenInfoText.text = "For the " + GoldenStage + "nd Round, you have to choose \r\n the
correct color based on font color! \r\n You can do it!";
            m_GoldInfoScreen.SetActive(true); //Shows info and prepares for the next level
        }else if(GoldenStage == 2)
        {
            GoldenStage = 3;
            m_GoldenInfoText.text = "The fun starts now! \r\nFor the " + GoldenStage + "rd Round, you
have to press \r\n the correct color based on font color! \r\n Be careful, because the color's name \r\n
can trick you and make you choose wrong! \r\n Do you see a difference?";
            m_GoldInfoScreen.SetActive(true); //Shows info and prepares for the next level
        }else if (GoldenStage == 3)
        {
            //Handle the game over
            GoldenStage = 4;
            gameOver = true;
            isPaused = true;
            m_GoldenScoreText.text = "" + Gscore; //Shows the score of the round
            m_GoldResultScreen.SetActive(true); //Shows the result and prepares for the next level
        }
    }
```

Κώδικας 6 : Η μέθοδος NextGoldenStage()

```
    public void CloseGoldenResultScreen()
    {
        if (GoldenStage==1)
        {
            m_GoldInfoScreen.SetActive(false); //Set the high score screen to inactive as it is closed
now
            spinner.ResetRotation(); //Reset the white UI's rotation to match the time
            gameStartCountdown = true; //Starts the countdown for the game to start
        }
        else if (GoldenStage <= 3)
        {
            m_GoldenTimer = 20.0f; //Restart timer and resets it to 20 seconds
            GoldenSize = 0; //Resets the color's counter to 0 ! (Max = 40)
            spinner.ResetRotation(); //Reset the white UI's rotation to match the time
            m_GoldInfoScreen.SetActive(false); //Set the high score screen to inactive as it is closed
now
            isPaused = false; //The game will start now
            this.NewGolden(); //Next word
        }
        else
        {
            m_GoldResultScreen.SetActive(false); //Set the high score screen to inactive as it is closed
now
            m_TimerText.text = "0.0"; //Set the time to lock at 0
            this.GameOver(); //Game Over
        }
    }
```

Κώδικας 7: Η μέθοδος CloseGoldenResultScreen()

6.4 Βαθμολογία

Για την βαθμολογία χρησιμοποιείται μια κλάση απο την Unity, που ονομάζεται PlayerPrefs. Συνήθως χρησιμοποιείται για αποθήκευση ρυθμίσεων της εφαρμογής ή για βαθμολογία. Ο λόγος είναι οτι αποθηκεύονται πληροφορίες τοπικά σε αρχεία και είναι προσβάσιμα στον χρήστη ώστε να τα επεξεργαστεί και να «κλέψει» το παιχνίδι. Μπορεί να αποθηκεύσει μόνο ακέραιους, δεκαδικούς αριθμούς ή ένα string (=ακολουθία αλφαριθμητικών χαρακτήρων). Επίσης χρειάζεται και ένα κλειδί που απευθύνεται στην αποθηκευμένη μεταβλητή. Στην εφαρμογή την δικία μας στο GameManager.cs βρίσκονται δύο μεθόδοι αποθήκευσης και ανάκλησης βαθμολογίας.

```
        private void UpdateHighScore() {
        //Saves the high score to a local file
        if (gameType == GameType.GoldenStroop)
        {
            PlayerPrefs.SetInt("GoldenHighScore", GhighScore); //saves golden stroop highscore
        }
        else
        {
            PlayerPrefs.SetInt("HighScore1", highScore);//saves endless mode high score
        }
    }


        /// <summary>
        /// Gets the high score.
        /// Hanlde and logic here that you want to retrieve the players high score.
        /// </summary>
        private void GetHighScore() {
                if (PlayerPrefs.HasKey ("HighScore1")) {
                        //Gets the high score from a local file
                        highScore = PlayerPrefs.GetInt ("HighScore1"); //Get endless mode high score
            GhighScore = PlayerPrefs.GetInt("GoldenHighScore"); //Get golden stroop highscore


            m_HighScoreText.text = "Previous HighScore: \r\n" + "" + highScore; //Display the
        exisiting high score
                }
        }
```

Κώδικας 8 : Ο κώδικας για την ενημέρωση και ανάκληση του High Score

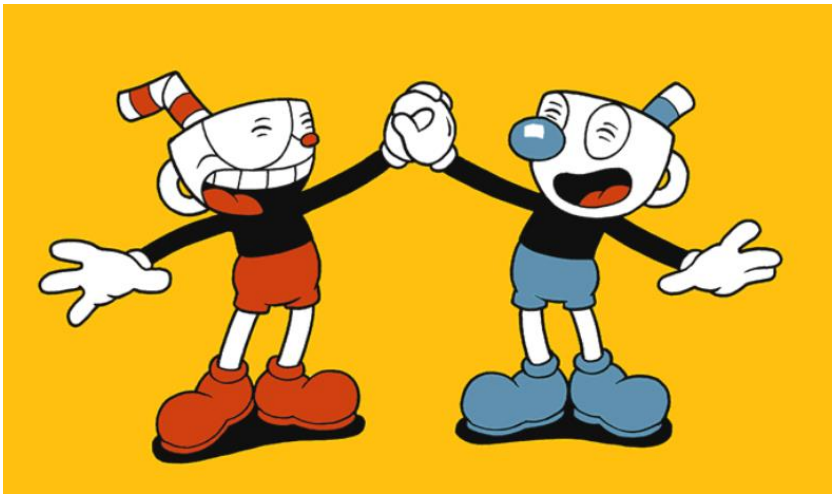
7 Συμπέρασμα

Όλες οι τρισδιάστατες πλατφόρμες ανάπτυξης παιχνιδιών προσφέρουν αμέτρητες δυνατότητες στον προγραμματιστή να δημιουργήσει με τον δικό του μοναδικό τρόπο ένα βιντεοπαιχνίδι όπως το έχει στο μυαλό του. Θα χρειαστεί και άλλα προγράμματα επεξεργασίας εικόνας, ήχου, προγραμματισμού χρονοδιαγράμματων και άλλων προγραμμάτων, αλλά το κύριο πρόγραμμα που θα υλοποιήσει το παιχνίδι θα είναι αυτές οι τρισδιάστατες πλατφόρμες.

Συγκεκριμένα εμείς εργαστήκαμε πάνω στην Unity και χάρης στο υλικό μέσω Youtube, Stack Overflow, Reddit και Unity Community μπορέσαμε να ξεπεράσουμε προβλήματα, να μάθουμε καινούργιες τεχνικές και τρόπους προγραμματισμού και τέλος να φτιάξουμε ένα πρόγραμμα όπως το θέλαμε. Όσο εξοικειώνεται κάποιος με το περιβάλλον της Unity τόσο πιο εύκολη είναι η δημιουργία και η πραγματοποίηση της ιδέας του.

Το φαινόμενο Stroop είναι ένα ενδιαφέρον φαινόμενο το οποίο το χρησιμοποιούν πολλοί διαφημιστές σήμερα, με τα πολλά λογότυπα των εταιριών τους που εμφανίζονται μπροστά μας. Με περαιτέρω αναβαθμίσεις στο πρόγραμμα θα μπορούσε να γίνει μια διασκεδαστική εφαρμογή στο Google Play ή ως ένα εργαλείο χορήγησης τέστ σε ένα κέντρο πειραματικής ψυχολογίας.

Τέλος θα θέλαμε να κλείσουμε αυτή την πυχιακή εργασία με τις εντυπώσεις μας, πάνω στις πλατφόρμες ανάπτυξης και ιδιαίτερα στην Unity, που ήταν φανταστικές. Σίγουρα θα χειριστούμε την Unity άμεσα στο μέλλον για την δημιουργία και την εκπαίδευσή μας πάνω σε άλλα project και δεν είναι καθόλου απίθανο να δούμε και στην πράξη τις δυνατότητες και των άλλων μηχανών βιντεοπαιχνιδιών.



Εικόνα 34 : Ο Cuphead και ο αδερφός του ο Mugman, πρωταγωνιστές στο ομόνυμο πολυβραβευμένο παιχνίδι «Cuphead» , που έχει δημιουργηθεί στην Unity.

8 Βιβλιογραφία

[1] Ιστορία των βιντεοπαιχνιδιών : https://en.wikipedia.org/wiki/History_of_video_games

[2] Ένα βίντεο για την ιστορία των βιντεοπαιχνιδιών απο τον χρήστη Ahoy : https://www.youtube.com/watch?v=GoyGlyrYb9c

[3] Οι κονσόλες βιντεοπαιχνιδιών : https://en.wikipedia.org/wiki/Video_game_console

[4] Οι μηχανές βιντεοπαιχνιδών : https://en.wikipedia.org/wiki/Game_engine

[5] Ένα άρθρο σχετικά με τις μηχανές βιντεοπαιχνιδιών : https://www.giantbomb.com/profile/michaelenger/blog/game-engines-how-do-they-work/101529/

[6] Πληροφορίες σχετικά με την Id tech : https://en.wikipedia.org/wiki/Id_Tech

[7] Πληροφορίες σχετικά με την Unreal : https://en.wikipedia.org/wiki/Unreal_Engine

[8] Λίστα βιντεοπαιχνιδιών που έχουν δημιουργηθεί με Unreal : https://en.wikipedia.org/wiki/List_of_Unreal_Engine_games

[9] Πληροφορίες σχετικά με την Unity : https://en.wikipedia.org/wiki/Unity_(game_engine)

[10] Η αρχική έρευνα «Studies of Interference in Serial verbal reactions» του J. Ridley Stroop: http://psychclassics.yorku.ca/Stroop/?fbclid=IwAR2m8WYY6bgrpeDRjpQna9ZeQFIwXY-ruFp7kD7muh-g52IUwihiZDOUSg8

[11] Η έρευνα «Half a Century of Research on the Stroop Effect: An Integrative Review» του Colin M. MacLeod : http://melaniestefan.net/MacLeod1991.pdf

[12] Η αρχική σελίδα της Unity : https://unity.com/

[13] Το πλήρες ενχειρίδιο χειρισμού της Unity: https://docs.unity3d.com/Manual/index.html

[14] Ένας οδηγός σχετικά με το αρχικό μενού της Unity : http://www.informit.com/articles/article.aspx?p=2464010&seqNum=2&fbclid=IwAR2MHzsvJFKFbd2gIu-QmdbbwPOOd4i6Q4JAQRg_gd1uCxcD4AOooZPPRMw

[15] Ένα site με προβλήματα και απαντήσεις για τις γλώσσες προγραμματισμού: https://stackoverflow.com/

[16] Το πρόγραμμα για επεξεργασία ήχου : https://www.audacityteam.org/

[17] Το πρόγραμμα για επεξεργασία εικόνας : https://www.gimp.org/

[18] Το πρόγραμμα για επεξεργασία της C# : https://visualstudio.microsoft.com/vs/

Assets που χρησιμοποιήθηκαν

[1] Stroop Effect by Cozy Cabin Studio : https://assetstore.unity.com/packages/templates/stroop-effect-game-template-90330

[2] 371 Simple Buttons by Nayrissa : https://assetstore.unity.com/packages/2d/gui/icons/371-simple-buttons-pack-97516

[3] UI Effects by mob-sakai : https://github.com/mob-sakai/UIEffect

[4] Unrealeased Game Music Pack by Cafofo : https://assetstore.unity.com/packages/audio/music/unreleased-game-music-pack-62626

9 Παράρτημα

AudioSourceManager.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

/// <summary>
/// Audio source manager.
/// Attach to an audio source in the scene to update the volume levels.
/// Any additional logic for volume control can go in here.
/// </summary>
public class AudioSourceManager : MonoBehaviour {

        private AudioSource audioSource; //The scenes audio source

        /// <summary>
        /// Awake this instance.
        /// </summary>
        void Awake() {

                audioSource = GetComponent<AudioSource> (); //Find the audio
source in the scene to use

                //Update the volume level for the scene
                audioSource.volume = GlobalVariables.VolumeLevel;
        }
    private void Update()
    {
        //Update the volume level for the scene
        audioSource.volume = GlobalVariables.VolumeLevel;
    }

    /// <summary>
    /// Plays the audio clip given.
    /// </summary>
    /// <param name="clipToPlay">Clip to play.</param>
    public void PlayAudioClip(AudioClip clipToPlay) {
```

ColorManager.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

/// <summary>
/// Color manager.
/// Basic version of the color manager for the game.
/// Uses a simple color and name of a color.
/// </summary>
public class ColorManager : MonoBehaviour {

        public ColorObject[] m_GameColors; //The color objects to be used in the game

        public Text m_ColorText; //Text of the color in the game

        private int currentColorIndex; //Stored the current color index selected
        private int maxIndexSize; //Stores the maximum size of the index to prevent
multiple lookups

        /// <summary>
```

```
        /// Start this instance.
        /// </summary>
        void Start() {
                maxIndexSize = m_GameColors.Length; //Set the max size reference

                this.GetNextColor (); //Get the next color to update the scene
        }

        /// <summary>
        /// Gets the next color object
        /// </summary>
        public void GetNextColor() {
                //We want to check for out of bounds in the index and prevent it
                if (currentColorIndex >= maxIndexSize -1) {
                        currentColorIndex = 0; //If the index would be out of bounds, set
it to the first index
                } else {
                        currentColorIndex++; //Otherwise we can go to the next index
                }

                //Set the color and text to the correct values
                m_ColorText.text = m_GameColors [currentColorIndex].m_ColorName;
                m_ColorText.color = m_GameColors [currentColorIndex].m_Color;
        }
}
```

ColorObject.cs

```
using System.Collections;
using UnityEngine;

/// <summary>
/// Color object.
/// Stores the basic info for building our color choices
/// </summary>
[System.Serializable]
public class ColorObject : MonoBehaviour {

        public Color32 m_Color; //The actually visible color to use
        public string m_ColorName; //The name of the color

}
```

MenuManager.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;

/// <summary>
/// Menu manager.
/// Manages the menu flow for the main menu
/// </summary>
public class MenuManager : MonoBehaviour {

        /// <summary>
        /// Loads the game.
        /// Give a game type number to load the correct linked game scene
        /// </summary>
        /// <param name="gameType">Game type.</param>
        public void LoadGame(int gameType) {
                switch (gameType) {
                case 0:
                        GlobalVariables.GameToPlay = 0; ///Endless Mode

                        break;
                ///case 1:
                ///     GlobalVariables.GameToPlay = 1; Reserved For Future Game Mode
                ///     break;
        case 2:
            GlobalVariables.GameToPlay = 2; ///Golden Stroop
            break;
        default:
                    break;
                }

                SceneManager.LoadScene (GlobalVariables.GameScene); //Loads the game scene
given
        }
}
```

InfoPopup.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class InfoPopup : MonoBehaviour
{
    public GameObject InfoUI;

    public void Start()
    {
        InfoUI.SetActive(false);
    }
    public void OnMouseOver()
    {
        InfoUI.SetActive(true);
    }
    public void OnMouseExit()
    {
        InfoUI.SetActive(false);

    }
}
```

OptionManager.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

/// <summary>
/// Options manager.
/// Controls the options menu in the main scene.
/// </summary>
public class OptionsManager : MonoBehaviour {

        public GameObject m_OptionsMenu; //The options menu canvas object
    public GameObject m_Info; //The info canvas object
    public GameObject m_Highscore; //The info canvas object
    public GameObject m_Lights; //The Lights object
    public Text m_VersionNumberText; //Our version number to display
    private int highScore; //Our highscore for endless mode from local files
    private int GhighScore; //Our highscore for Golden Stroop mode from local files
    public Text m_HighScoreText; //The variable that saves Highscore for Endless Mode
    public Text m_GHighScoreText; //The variable that saves Highscore for Golden Stroop
Mode

    /// <summary>
    /// Start this instance.
    /// </summary>
    void Start() {
                m_OptionsMenu.SetActive (false); //Option menu will not display by
default, so disable it to start
        m_Info.SetActive(false); //Option menu will not display by default, so disable it
to start
    }

        /// <summary>
        /// Shows the options menu.
        /// </summary>
        public void ShowOptionsMenu() {
        m_Info.SetActive(false); //Closes Info if it is open
        m_Highscore.SetActive(false); //Closes high score screen if it is open
        m_OptionsMenu.SetActive (true); //Set the options menu to enabled

                m_VersionNumberText.text = GlobalVariables.VersionNumber; //Update our
version number from the global variables
        }

        /// <summary>
        /// Closes the options menu.
        /// </summary>
        public void CloseOptionsMenu() {

                m_OptionsMenu.SetActive (false); //Disable the options menu
        }
    /// <summary>
        /// Shows the infos.
        /// </summary>
    public void ShowInfos()
    {
        m_OptionsMenu.SetActive(false); //Closes Option menu if it is open
        m_Highscore.SetActive(false); //Closes high score screen if it is open
        m_Info.SetActive(true); //Set informations to enabled

    }

    /// <summary>
    /// Closes the infos
    /// </summary>
    public void CloseInfos()
    {
```

```
        m_Info.SetActive(false); //Disable informations
    }
    public void ShowHS()
    {
        m_OptionsMenu.SetActive(false); //Closes Option menu if it is open
        m_Info.SetActive(false); //Closes informations if it is open
        highScore = PlayerPrefs.GetInt("HighScore1"); //Gets the highscore for Endless
Mode from local files
        GhighScore = PlayerPrefs.GetInt("GoldenHighScore"); //Gets the highscore for
Golden Stroop from local files
        m_HighScoreText.text = "" + highScore; //Changes the text of UI.Button.text to
show highscore of Endless mode
        m_GHighScoreText.text = "" + GhighScore; //Changes the text of UI.Button.text to
show highscore of Golden Stroop mode
        m_Highscore.SetActive(true); //Set highscore screen ON
        ///<summary>
        ///Here is the logic for animated Lights! Γεννηθήτω φως !
        ///</summary>
        m_Lights.SetActive(true); //Set Lights ON
    }

    /// <summary>
    /// Closes the infos
    /// </summary>
    public void CloseHS()
    {
        m_Highscore.SetActive(false); //Disable High Score Screen
    }

    /// <summary>
    /// Clears the saved data.
    /// </summary>
    public void ClearSavedData() {
                //Clear all the player pref data and save it
                PlayerPrefs.DeleteAll ();
                PlayerPrefs.Save ();
        }
    /// <summary>
    /// Exits the game.
    /// </summary>
    public void doExit() {
        //Exits the game
        Application.Quit();
    }

}
```

SoundManager.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

/// <summary>
/// Sound manager.
/// Controls the sound volume levels for the game and menu
/// </summary>
public class SoundManager : MonoBehaviour {

	private float volumeLevel = 1.0f; //The current volume level

	private float maxVolumeLevel = 1.0f; //The max volume level
	private float minVolumeLevel = 0.0f; //The minimum volume level

	/// <summary>
	/// Raises the awake event.
	/// </summary>
	void OnAwake() {
		//The sound manager will persist across all scenes in this project
		DontDestroyOnLoad (this);
	}

	/// <summary>
	/// Toggles the volume.
	/// </summary>
	public void VolumeToggle(bool volumeEnabled) {
		//If the volume is enables, disable it, and vice-versa
		if (volumeEnabled) {
			volumeLevel = maxVolumeLevel;
		} else {
			volumeLevel = minVolumeLevel;
		}

		this.UpdateVolume(); //Update the global variable for the sound level
	}

	/// <summary>
	/// Updates the volume level.
	/// </summary>
	private void UpdateVolume() {
		GlobalVariables.VolumeLevel = volumeLevel; //Set the global variable to
reflect the volume level to use
	}
}
```

Spinner.cs

```
using UnityEngine;
using System.Collections;

/// <summary>
/// Spinner.
/// Rotates and object at given rotation number, axis (X,Y,Z) and forward or backward
motion
/// </summary>
public class Spinner : MonoBehaviour {

        public float m_RotationsPerMin = 3.0f; //The number of full rotations to do per
minute
    public bool isPaused;
    public Quaternion originalRotationValue;
    public enum RotationAngle { X, Y, Z }; //What axis to rotate on
        public RotationAngle m_RotationWay;

        public enum Direction { Forward, Backward }; //Do you want to go clockwise or
counter-clockwise
        public Direction m_RotationDirection;

        private float directionModifier; //Modifier for controlling the rotation way

    void Start()
    {
        originalRotationValue = transform.rotation; // save the initial rotation
        isPaused = true;
    }
    /// <summary>
    /// Update this instance.
    /// </summary>
    void Update () {

        //Check to see which direction has been given and update the modifier
        if (m_RotationDirection == Direction.Backward) {
                        directionModifier = -1;
                }
                else {
                        directionModifier = 1;
                }
        if (isPaused==false)
        {
            //This switch will rotate the object on the correct axis depending on the
inputs given in the inspector
            switch (m_RotationWay)
            {
                case RotationAngle.X:
                    this.transform.Rotate(6.0f * m_RotationsPerMin * Time.deltaTime *
directionModifier, 0.0f, 0.0f);
                    break;
                case RotationAngle.Y:
                    this.transform.Rotate(0.0f, 6.0f * m_RotationsPerMin * Time.deltaTime
* directionModifier, 0.0f);
                    break;
                case RotationAngle.Z:
                    this.transform.Rotate(0.0f, 0.0f, 6.0f * m_RotationsPerMin *
Time.deltaTime * directionModifier);
                    break;
            }

        }
    }
    /// <summary>
    /// A method to set the rotation to the best value for Golden Test.
    /// (Needs testing)
    /// </summary>
    public void GoldenRotation()
    {
```

```
        m_RotationsPerMin = 0.75f;
    }
    /// <summary>
    /// A method that resets rotations to gameobject.
    /// </summary>
    public void ResetRotation()
    {
        this.transform.rotation = Quaternion.Lerp(transform.rotation,
originalRotationValue, Time.time * 200.0f);
    }
    /// <summary>
    /// A method that decreases the rotation everytime.
    /// </summary>
    public void DecreaseRotation(float _decreaseRotation)
    {
        m_RotationsPerMin = m_RotationsPerMin + _decreaseRotation;
    }


    /// <summary>
    /// A method that stops the rotation.
    /// </summary>
    public void StopRotation()
    {
        isPaused = true;
    }
    /// <summary>
    /// A method that starts the rotation.
    /// </summary>
    public void StartRotation()
    {
        isPaused = false;
    }
}
```

KeyPressButton.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class KeypressButton : MonoBehaviour
{
    public KeyCode _key;
    private Button _button;

    private void Awake()
    {
        _button = GetComponent<Button>();
    }

    // Update is called once per frame
    void Update()
    {
        if (Input.GetKeyDown(_key))
        {
            _button.onClick.Invoke();
        }
    }
}
```

GameManager.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.SceneManagement;

/// <summary>
/// Game manager.
/// Handles the logic to play the Stroop quick selection game.
/// </summary>
public class GameManager : MonoBehaviour {

        private int score; //Current score
    private int Gscore; //Current score
    private int highScore; //Save (local) high score
    private int GhighScore; //Save (local) high score for golden stroop

    public float m_RoundTimer = 3.0f; //The timer for each question
        public float m_LowestTime = 2.0f; //The minimum time to give the player to answer
        public float m_TimeDecreasePerAnswer = 0.05f; //Time to decrease upon right answer
    private float m_RotationDecreasePerAnswer = 0.033f; //A variable to slow rotation
    private float currentTimer; //To track the time remaining in the round

        private bool isPaused; //Control for pausing the game or going to the menu

        private bool gameOver; //Control for the game over flag

        private bool gameStartCountdown; //Startup for start of new game
        public float m_GameStartTimer = 5.0f; //Timer shown before the game starts
    public float m_GoldenTimer = 20.0f; //Timer shown before the Golden Stroop starts
    public float m_wrongGolden = 0.5f; //The decreasing amount of time per wrong answer

    private ColorManager colorManager; //The instance of the color manager in the scene
        private ColorShapeManager colorShapeManager; //The instance of the color shape
manager in the scene
        private AudioSourceManager audioSourceManager; //The instance of the audio manager
in the scene
    private Spinner spinner; //The instance of spinner in the scene

    private bool isColorMatch; //Flag for knowing when the color and word match
    private int GoldenColor = 0; //The Random word
    private int GoldenStage ; //The stage counter for the 3 stages if Stroop experiment
    private int GoldenSize; //The amount of examples the program provides to user
    public int maxGoldenSize; // Max amount for examples

    //Enumerator for determining the game type.
    //Add any new game types here once the logic has been established.
    private enum GameType
        {
                ColorWord,
                ColorShape,
        GoldenStroop,
                Menu
        }
        private GameType gameType; //Initialize the instance of our enumerator for use in
logic


        #region UI Elements
        public Text m_ColorText; //The text to show for determining the right answer

        public Image m_Shape; //The shape to show for determining the correct answer

        public Button m_Button1; //Button choice 1
        public Button m_Button2; //Button choice 2
    public Button m_Button3; //Button choice 3
    public Button m_Button4; //Button choice 4
    public Button m_Button5; //Button choice 5

    public Text m_TimerText; //The game timer
```

```
        public Text m_HighScoreText; //The text display for the high score

        public Text m_NewHighScoreText; //New high score to show on the high score screen
    public Text m_GoldenInfoText; //The text that shows instruction for the next round
    public Text m_GoldenScoreText; //New high score to show on the high score screen
    public Text m_CountdownText; //The text display for the countdown

        public GameObject m_GameOverScreen; //The screen to show on a game over
        public GameObject m_HighScoreScreen; //The screen to show on a new high score
    public GameObject m_GoldResultScreen; //The screen to show on Golden Stroop test
completion
    public GameObject m_GoldInfoScreen; //The screen to show on Golden stage completion and
info for the next round

    public GameObject m_CountDownScreen; //The screen to show for the initial countdown
        #endregion

        public AudioClip m_CorrectAnswerClip; //Sound to play on the correct answer given
    public AudioClip m_WrongAnswerClip; //Sound to play on the correct answer given
    public AudioClip m_GameOverClip; //Sound to play on a game over

        /// <summary>
        /// Start this instance.
        /// All things that need to be assigned on the scene start can go in here.
        /// </summary>
        void Start() {
                this.SetupGame ();
                this.GetHighScore (); //Get the high score for comparison later

                m_GameOverScreen.SetActive (false); //Disable the game over screen for now

                m_CountDownScreen.SetActive (true); //Show the countdown screen to start
        }

        /// <summary>
        /// Fixed update is used in this project to keep the timing the same across all
scripts.
        /// This is especially important for our timers so that they do not stutter or lose
count.
        /// </summary>
        void FixedUpdate() {

                if (gameType == GameType.ColorWord) {

                        //Check to see if the game has just started so that we can run the
initial countdown until start
                        if (gameStartCountdown) {
                                m_GameStartTimer -= Time.deltaTime; //Start decreasing time
towards 0

                                m_CountdownText.text = "" + m_GameStartTimer.ToString
("F1");

                                if (m_GameStartTimer <= 0.0f) {
                                        m_CountDownScreen.SetActive (false); //Disable the
countdown timer screen
                                        gameStartCountdown = false; //The countdown has
ended, so toggle this off
                                        isPaused = false; //The game will start now
                    spinner.StartRotation();
                                        this.NewWord ();

                                }
                        }

                        //Check to see if the game is paused right now, if not, do the
countdown
                        if (!isPaused) {
                                currentTimer -= Time.deltaTime; //Delta time will help
prevent time loss or skipping

                                m_TimerText.text = "" + currentTimer.ToString ("F1");

                                //Check for a out of time count on the answer
```

```
                                if (currentTimer <= 0.0f) {
                                        isPaused = true; //Pause the game as it is now
over

                                        m_TimerText.text = "0.0"; //Set the time to lock
at 0

                                        //Handle the game over
                                        gameOver = true;
                                        isPaused = true;
                    spinner.StopRotation();
                    this.GameOver ();
                                }
                        }
                }

                if (gameType == GameType.ColorShape) {

                        //Check to see if the game has just started so that we can run the
initial countdown until start
                        if (gameStartCountdown) {
                                m_GameStartTimer -= Time.deltaTime; //Start decreasing time
towards 0

                                m_CountdownText.text = "" + m_GameStartTimer.ToString
("F1");

                                if (m_GameStartTimer <= 0.0f) {
                                        m_CountDownScreen.SetActive (false); //Disable the
countdown timer screen
                                        gameStartCountdown = false; //The countdown has
ended, so toggle this off
                                        isPaused = false; //The game will start now
                                        this.NewShape ();
                                }
                        }

                        //Check to see if the game is paused right now, if not, do the
countdown
                        if (!isPaused) {
                                currentTimer -= Time.deltaTime; //Delta time will help
prevent time loss or skipping

                                m_TimerText.text = "" + currentTimer.ToString ("F1");

                                //Check for a out of time count on the answer
                                if (currentTimer <= 0.0f) {
                                        isPaused = true; //Pause the game as it is now
over

                                        m_TimerText.text = "0.0"; //Set the time to lock
at 0

                                        //Handle the game over
                                        gameOver = true;
                                        isPaused = true;

                                        this.GameOver ();
                                }
                        }
                }
```

```
        if (gameType == GameType.GoldenStroop)
        {
            //Check to see if the game has just started so that we can run the initial
countdown until start
            if (gameStartCountdown)
            {
                m_GameStartTimer -= Time.deltaTime; //Start decreasing time towards 0

                m_CountdownText.text = "" + m_GameStartTimer.ToString("F1");

                if (m_GameStartTimer <= 0.0f)
                {
                    m_CountDownScreen.SetActive(false); //Disable the countdown timer
screen
                    gameStartCountdown = false; //The countdown has ended, so toggle this
off
                    isPaused = false; //The game will start now
                    spinner.StartRotation();
                    this.NewGolden();
                }

            }

            //Check to see if the game is paused right now, if not, do the countdown
            if (!isPaused)
            {
                m_GoldenTimer -= Time.deltaTime; //Delta time will help prevent time loss
or skipping
                m_TimerText.text = "" + m_GoldenTimer.ToString("F1");
                //Check if the sample was given to Player

                if (GoldenSize >= maxGoldenSize)
                {
                    isPaused = true;
                    this.NextGoldenStage();
                }
                //Check for a out of time count on the answer
                if (m_GoldenTimer <= 0.0f)
                {
                    isPaused = true; //Pause the game as it is now over

                    m_TimerText.text = "0.0"; //Set the time to lock at 0

                    //Show results and enter next stage
                    this.NextGoldenStage();
                }
            }
        }

    }

    public void SetupGame() {
                colorManager = FindObjectOfType<ColorManager>() as ColorManager; //Get the
Color Manager instance for the scene to reference
                colorShapeManager = FindObjectOfType<ColorShapeManager>() as
ColorShapeManager; //Get the instance of the manager
                audioSourceManager = FindObjectOfType<AudioSourceManager>() as
AudioSourceManager; //Get the only instance of the audio source manager in the scene
        spinner = FindObjectOfType<Spinner>() as Spinner; //Get the only instance of the
audio source manager in the scene

        currentTimer = m_RoundTimer; //Assign the total time available per answer as it
will be decreasing

                isPaused = true; //The game will need to do a countdown before starting so
pause it
                gameStartCountdown = true; //Game will start so load the countdown timer
                gameOver = false; //We just started the game so it will not be over yet

                this.SetGameType (GlobalVariables.GameToPlay);
        }

        public void LeaveGame() {
                //Remove the reference to the managers as we are leaving the scene
```

```
                colorManager = null;
                colorShapeManager = null;

                currentTimer = m_RoundTimer; //Assign the total time available per answer
as it will be decreasing

                isPaused = true; //The game will need to do a countdown before starting so
pause it
                gameStartCountdown = false; //We want to wait for the player to select the
game type before starting the countdown
                gameOver = true; //The game is now over

                gameType = GameType.Menu;

                SceneManager.LoadScene ("MainMenu"); //Load which will be the main menu
scene
        }

        /// <summary>
        /// Sets the type of the game.
        /// </summary>
        /// <param name="gameNumber">Game number.</param>
        public void SetGameType(int gameNumber) {
                //Using a switch statement will allow for streamlined setting of the game
type
                //from outside the class.
                switch (gameNumber) {
                case 0:
                        gameType = GameType.ColorWord;
                        m_Shape.enabled = false;
                        break;
                case 1:
                        gameType = GameType.ColorShape;
                        m_Shape.enabled = true;
                        break;
        case 2:
            gameType = GameType.GoldenStroop;
                m_Shape.enabled = false;
                GoldenSize = 0;
                GoldenStage = 1;
                Gscore = 0;
                spinner.GoldenRotation();
                m_Button1.gameObject.SetActive(false); //Hides Endless Mode UI and shows
Golden stroop's UI
                m_Button2.gameObject.SetActive(false); //Hides Endless Mode UI and shows
Golden stroop's UI
                m_HighScoreText.gameObject.SetActive(false); //Hides Endless Mode UI and
shows Golden stroop's UI
                m_Button3.gameObject.SetActive(true); //Hides Endless Mode UI and shows
Golden stroop's UI
                m_Button4.gameObject.SetActive(true); //Hides Endless Mode UI and shows
Golden stroop's UI
                m_Button5.gameObject.SetActive(true); //Hides Endless Mode UI and shows
Golden stroop's UI
                m_GoldenInfoText.text = "For the " + GoldenStage + "st Round you have to
choose \r\n the correct color based on name! \r\n Easy right?";
                m_GoldInfoScreen.SetActive(true); //Shows info and prepares for the next
level
                gameStartCountdown = false;
                isPaused = true;
            break;
        default:
                    gameType = GameType.ColorWord;
                    break;
                }
        }

        /// <summary>
        /// Gets a new word, color, and shape.
        /// </summary>
        public void NewShape() {
                //We need a color and word and don't want them to be the same, so we get
two random references
                //in the array which will be used to determine the question and answer
```

```
                int randomWord = Random.Range (0, colorShapeManager.m_GameColors.Length);
                int randomColor = Random.Range (0, colorShapeManager.m_GameColors.Length);
                int randomShape = Random.Range (0, colorShapeManager.m_Shapes.Length);
                int randomDisplayShape = Random.Range (0,
colorShapeManager.m_Shapes.Length);

                /*
                 * If random word matches the random color
                 * If random shape matches random display shape
                 * Match
                 */

                //Set the text to be the word and the random shape
                m_ColorText.text = colorShapeManager.m_GameColors [randomWord].m_ColorName
+ " " +
                        colorShapeManager.m_Shapes[randomShape].name;
                //Set the color to be the random color
                m_ColorText.color = colorShapeManager.m_GameColors [randomColor].m_Color;
                //Set the display shape to the random display shape
                m_Shape.sprite = colorShapeManager.m_Shapes [randomDisplayShape];
                m_Shape.color = colorShapeManager.m_GameColors [randomColor].m_Color;

                if (randomWord == randomColor && randomShape == randomDisplayShape) {
                        isColorMatch = true;
                } else {
                        isColorMatch = false;
                }
        }
```

```
        /// <summary>
        /// Gets a new word and color.
        /// </summary>
        public void NewWord() {
        //We need a color and word and don't want them to be the same, so we get two random
references
        //in the array which will be used to determine the question and answer
        int randomWord = Random.Range (0, colorManager.m_GameColors.Length);
                int randomColor = Random.Range (0, colorManager.m_GameColors.Length);

                //Set the color and text accordingly
                m_ColorText.text = colorManager.m_GameColors [randomWord].m_ColorName;
                m_ColorText.color = colorManager.m_GameColors [randomColor].m_Color;

                //Check to see if the chosen array is the same result for a mtach
                if (randomWord == randomColor) {
                        isColorMatch = true;

                } else {
                        //No match, so set the flag to false
                        isColorMatch = false;
                }
        }
    /// <summary>
    /// Gets a new Golden Stroop test
    /// </summary>
    public void NewGolden()
    {
        //We need a color and word and don't want them to be the same, so we get two random
references
        //in the array which will be used to determine the question and answer
        int randomWord = Random.Range(0, colorManager.m_GameColors.Length);
        int randomColor = Random.Range(0, colorManager.m_GameColors.Length);
        if (GoldenStage == 1)
        {
            m_ColorText.text = colorManager.m_GameColors[randomWord].m_ColorName;  //Gets a
random color's name and shows on screen
            m_ColorText.color = Color.black; //Sets the font's color to black
```

```
            GoldenColor = randomWord; //Flag for Check answer

        }
        else if (GoldenStage == 2)
        {
            m_ColorText.text = "XXXX"; //The text that appears instead of color's name
            m_ColorText.color = colorManager.m_GameColors[randomColor].m_Color; //Gets a
random font color and shows on screen along with text
            GoldenColor = randomColor; //Flag for Check answer
        }
        else
        {
            m_ColorText.text = colorManager.m_GameColors[randomWord].m_ColorName; //Gets a
random color's name and shows on screen
            m_ColorText.color = colorManager.m_GameColors[randomColor].m_Color; //Gets a
random font color and shows on screen along color's name
            GoldenColor = randomWord; //Flag for Check answer
        }
    }

    /// <summary>
    /// Checks the answer.
    /// Button sending "0" will be true.
    /// Button sending "1" will be false.
    /// Compare the answer to a match, which will be true.
    /// Otherwise the answer will be false.
    /// </summary>
    /// <param name="buttonResult">Button result.</param>
    public void CheckAnswer(int buttonResult) {

                bool IsCorrect = false;
        if (gameType == GameType.GoldenStroop) {
            //Check to ensure that the game is not paused as this would allow cheating
            if (!isPaused)
            {
                /// <summary>
                /// 0 = Red , 1 = Blue  , 2 = Green
                /// </summary>

                GoldenSize++;

                //Is match and "0" = right
                //Is not a match and "1" = right
                //Otherwise = wrong
                if (GoldenColor==0 && buttonResult == 0)
                {
                    IsCorrect = true;
                }
                else if (GoldenColor == 1 && buttonResult == 1)
                {
                    IsCorrect = true;
                }
                else if (GoldenColor == 2 && buttonResult == 2)
                {
                    IsCorrect = true;
                }
                else
                {
                    IsCorrect = false;
                }
                //If the player got the answer correct add to their Golden score
                if (IsCorrect == true)
                {
                    Gscore++;
                    audioSourceManager.PlayAudioClip(m_CorrectAnswerClip); //Play the
correct answer clip
                    this.NewGolden();
                    IsCorrect = false;
                }
                else
                {
                    //Player got it wrong, so decrease (x)'' from Golden timer
                    m_GoldenTimer = m_GoldenTimer - m_wrongGolden;
                    audioSourceManager.PlayAudioClip(m_WrongAnswerClip);
```

```
                    this.NewGolden();
                }
            }
        }
        //Here if-statement is to check the gametype and the corresponding checkAnser
        else{
            //Check to ensure that the game is not paused as this would allow cheating
            if (!isPaused)
            {
                //Is match and "0" = right
                //Is not a match and "1" = right
                //Otherwise = wrong
                if (isColorMatch & buttonResult == 0)
                {
                    IsCorrect = true;
                }
                else if (!isColorMatch & buttonResult == 1)
                {
                    IsCorrect = true;
                }
                else
                {
                    IsCorrect = false;
                }

                //If the player got the answer correct add to their score and decrease the
timer
                if (IsCorrect)
                {
                    score++;
                    spinner.ResetRotation();
                    spinner.StartRotation();
                    audioSourceManager.PlayAudioClip(m_CorrectAnswerClip); //Play the
correct answer clip

                    //Check to make sure that we aren't going below the minimum time given
to answer
                    if (m_RoundTimer - (m_TimeDecreasePerAnswer * score) > 1.5)
                    {
                        currentTimer = m_RoundTimer - (m_TimeDecreasePerAnswer * score);
                    }
                    else
                    {
                        currentTimer = m_LowestTime; //Set to the lowest time if below or
at the limite of reduction
                    }

                    if (gameType == GameType.ColorWord)
                    {
                        this.NewWord();
                        if (currentTimer != m_LowestTime)
                        {
                            spinner.DecreaseRotation(m_RotationDecreasePerAnswer);
                        }
                        spinner.StartRotation();
                    }
                    else
                    {
                        this.NewShape();
                    }

                }
                else
                {
                    //Player got it wrong, so go to the game over handling
                    gameOver = true;
                    isPaused = true;
                    spinner.StopRotation();
                    this.GameOver();
                }
            }
        }
        }
```

```
    ///<summary>
    ///Move to the next Golden Stroop scene
    ///</summary>
    public void NextGoldenStage()
    {
        //Checks which stage we are on and make transition between stages.
        if (GoldenStage == 1 )
        {
            GoldenStage=2;
            m_GoldenInfoText.text = "For the " + GoldenStage + "nd Round, you have to
choose \r\n the correct color based on font color! \r\n You can do it!";
            m_GoldInfoScreen.SetActive(true); //Shows info and prepares for the next level
        }else if(GoldenStage == 2)
        {
            GoldenStage = 3;
            m_GoldenInfoText.text = "The fun starts now! \r\nFor the " + GoldenStage + "rd
Round, you have to press \r\n the correct color based on font color! \r\n Be careful,
because the color's name \r\n can trick you and make you choose wrong! \r\n Do you see a
difference?";
            m_GoldInfoScreen.SetActive(true); //Shows info and prepares for the next level
        }else if (GoldenStage == 3)
        {
            //Handle the game over
            GoldenStage = 4;

            gameOver = true;
            isPaused = true;
            m_GoldenScoreText.text = "" + Gscore; //Shows the score of the round
            m_GoldResultScreen.SetActive(true); //Shows the result and prepares for the
next level

        }

    }

        /// <summary>
        /// Game over handling.
        /// Any updates to the game logic can be done here when the player loses.
        /// </summary>
        public void GameOver() {

                //Check again for game over
                if (gameOver) {

                        audioSourceManager.PlayAudioClip (m_GameOverClip); //Play the game
over sound clip

                        //If the current score is larger than the high score we want to
update the high score
                        if (score > highScore) {
                                highScore = score; //Set the high score
                                this.UpdateHighScore (); //Update it to save file
                                m_HighScoreScreen.SetActive (true); //Set the high score
screen to active so we can see it
                                m_NewHighScoreText.text = "" + highScore; //Display the new
high score
                        }
            if (Gscore > GhighScore)
            {
                GhighScore = Gscore; //Set the high score
                this.UpdateHighScore(); //Update it to save file
            }
            spinner.StopRotation();
                        this.m_GameOverScreen.SetActive (true); //Enable the game over
screen
                }
        }

        /// <summary>
        /// Retry the current game.
        /// </summary>
```

```
        public void Retry() {
                SceneManager.LoadScene (SceneManager.GetActiveScene ().buildIndex);
//Reload the current scene
        }

        /// <summary>
        /// Returns to main menu.
        /// </summary>
        public void ReturnToMainMenu() {
                SceneManager.LoadScene (GlobalVariables.MenuScene); //Return to the main
menu scene by loading it
        }

        /// <summary>
        /// Updates the high score.
        /// Handle any logic here that you want to update scores.
        /// </summary>
        private void UpdateHighScore() {
        //Saves the high score to a local file
        if (gameType == GameType.GoldenStroop)
        {
            PlayerPrefs.SetInt("GoldenHighScore", GhighScore); //saves golden stroop
highscore
        }
        else
        {
            PlayerPrefs.SetInt("HighScore1", highScore);//saves endless mode high score
        }
    }

        /// <summary>
        /// Gets the high score.
        /// Hanlde and logic here that you want to retrieve the players high score.
        /// </summary>
        private void GetHighScore() {
                if (PlayerPrefs.HasKey ("HighScore1")) {
                        //Gets the high score from a local file
                        highScore = PlayerPrefs.GetInt ("HighScore1"); //Get endless mode
high score
            GhighScore = PlayerPrefs.GetInt("GoldenHighScore"); //Get golden stroop
highscore

            m_HighScoreText.text = "Previous HighScore: \r\n" + "" + highScore; //Display
the exisiting high score
                }
        }

        /// <summary>
        /// Closes the high score screen.
        /// </summary>
        public void CloseHighScoreScreen() {
                m_HighScoreScreen.SetActive (false); //Set the high score screen to
inactive as it is closed now
        }
    /// <summary>
    /// Closes the score screen and moves to next stage.
    /// </summary>
    public void CloseGoldenResultScreen()
    {
        if (GoldenStage==1)
        {
            m_GoldInfoScreen.SetActive(false); //Set the high score screen to inactive as
it is closed now
            spinner.ResetRotation(); //Reset the white UI's rotation to match the time
            gameStartCountdown = true; //Starts the countdown for the game to start
        }
        else if (GoldenStage <= 3)
        {
            m_GoldenTimer = 20.0f; //Restart timer and resets it to 20 seconds
            GoldenSize = 0; //Resets the color's counter to 0 ! (Max = 40)
            spinner.ResetRotation(); //Reset the white UI's rotation to match the time
            m_GoldInfoScreen.SetActive(false); //Set the high score screen to inactive as
it is closed now
            isPaused = false; //The game will start now
```

```
            this.NewGolden(); //Next word
        }
        else
        {
            m_GoldResultScreen.SetActive(false); //Set the high score screen to inactive as
it is closed now
            m_TimerText.text = "0.0"; //Set the time to lock at 0
            this.GameOver(); //Game Over
        }
    }

}
```