ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ – ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Πρόγραμμα Μεταπτυχιακών Σπουδών

« ΜΠΣ ΠΡΟΗΓΜΕΝΑ ΣΥΣΤΗΜΑΤΑ ΠΛΗΡΟΦΟΡΙΚΗΣ - ΑΝΑΠΤΥΞΗ ΛΟΓΙΣΜΙΚΟΥ ΚΑΙ ΤΕΧΝΗΤΗΣ ΝΟΗΜΟΣΥΝΗΣ »

Μεταπτυχιακή Διατριβή

Τίτλος Διατριβής	**ΑΝΑΠΤΥΞΗ ΠΑΙΧΝΙΔΙΟΥ ΑΤΕΡΜΟΝΗΣ ΔΙΑΔΡΟΜΗΣ ΓΙΑ ΚΙΝΗΤΕΣ ΣΥΣΚΕΥΕΣ ΜΕ ΔΥΝΑΤΟΤΗΤΑ ΕΠΙΚΟΙΝΩΝΙΑΣ ΜΕ ΑΠΟΜΑΚΡΥΣΜΕΝΗ ΒΑΣΗ ΔΕΔΟΜΕΝΩΝ** **DEVELOPMENT OF AN ENDLESS RUNNER VIDEO GAME FOR MOBILE PHONES WITH COMMUNICATION CAPABILITIES WITH REMOTE DATABASE**
Ονοματεπώνυμο Φοιτητή	**Στιβακτάς Παναγιώτης**
Πατρώνυμο	**Ιωάννης**
Αριθμός Μητρώου	**ΜΠΣΠ21047**
Επιβλέπων	**Θεμιστοκλής Παναγιωτόπουλος, Καθηγητής**

Τριμελής Εξεταστική Επιτροπή

Θεμιστοκλής Παναγιωτόπουλος Καθηγητής	Άγγελος Πικράκης Επίκουρος Καθηγητής	Ιωάννης Τασούλας Επίκουρος Καθηγητής

Περιεχόμενα

Περίληψη

Η παρούσα διπλωματική, έχει ως στόχο την δημιουργία ενός mobile παιχνιδιού τύπου «endless runner», κατασκευασμένο σε Unity , το οποίο θα έχει τη δυνατότητα να επικοινωνεί ανάμεσα σε συσκευές. κατά τη διάρκεια του παιχνιδιού ο παίκτης οδηγεί ένα αυτοκίνητο και έχει ως σκοπό να αποφύγει την σύγκρουση με τα εμπόδια που θα βρει στο δρόμο. Όσο περισσότερα εμποδια αποφεύγουν τόσο αυξάνεται και το σκορ του. Με τη χρήση της διαδικτυακής βάσης δεδομένων Google firebase ο κάθε χρήστης θα μπορεί να βλέπει τα σκορ των υπολοίπων παικτών, αλλά και να ενημερώνεται για το ποιος έχει την καλύτερη επίδοση μέσω του online πίνακα που ανανεώνεται με κάθε παιχνίδι.

Αντικείμενο της διπλωματικής εργασίας

Αντικείμενο της διπλωματικής εργασίας είναι η ανάπτυξη ενός παιχνιδιού που να είναι ευχάριστο για τους παίκτες του, αλλά και να τους προσφέρει την δυνατότητα μιας σχετικής επικοινωνίας μεταξύ χρηστών. Για την βέλτιστη κατανόηση του αντικειμένου και την δημιουργία μιας εφαρμογής ευπαρουσίαστης και λειτουργικής μελετήθηκαν προγραμματιστικά εργαλεία όπως :

- Βάσεις δεδομένων
- Google Firebase
- Unity 2d
- Γλώσσα προγραμματίσου C#
- Το πλήθος βιβλιοθηκών της C# που σχετίζονται με την κατασκευή παιχνιδιών

Αυτή η εκ βαθέου κατανόηση επέτρεψε τον σχεδιασμό του προγράμματος, ώστε να είναι εύχρηστο και λειτουργικό.

Εισαγωγή

Η εργασία αποτελείται από ένα παιχνίδι τύπου (endless runner) για κινητές συσκευές, δηλαδή ένα πρόγραμμα στο οποίο ο παίκτης χειρίζεται έναν χαρακτήρα (στην προκειμένη περίπτωση ένα αυτοκίνητο), ο οποίος τρέχει σε πάνω σε ένα περιβάλλον που ανακυκλώνεται συνεχώς, χωρίς τη δυνατότητα να σταματήσει ή να επιβραδύνει.

Σκοπός του παιχνιδιού είναι η αποφυγή των εμποδίων που ξεπροβάλουν μπροστά από τον παίκτη και η επίτευξη όσο το δυνατόν μεγαλύτερου δυνατού σκορ. Το σκορ αυξάνεται όσο περνάει ο χρόνος χωρίς ο παίκτης να έχει χτυπήσει σε κάποιο εμπόδιο και φαίνεται στο πάνω μέρος της οθόνης. Στο τέλος του παιχνιδιού ο παίκτης μπορεί να δει τις επιδόσεις όλων των υπολοίπων χρηστών καθώς αυτές αποθηκεύονται σε μια βάση δεδομένων Microsoft Firebase.

ΜΕΡΟΣ ΠΡΩΤΟ

ΘΕΩΡΗΤΙΚΟ

Παιχνίδια για κινητές συσκευές

Κατά την τελευταία δεκαετία, τα smartphones έχουν γίνει αναπόσπαστο κομμάτι της καθημερινής ζωής και της εργασίας, προσφέροντας ισχυρή δυνατότητα επεξεργασίας και ευαίσθητες οθόνες που έχουν ανανεώσει την εμπειρία του παιχνιδιού. Η ευρεία διαθεσιμότητα των καταστημάτων εφαρμογών και η εμφάνιση δωρεάν εφαρμογών / παιχνιδιών έχουν διευκολύνει σημαντικά τη διανομή εφαρμογών. Σε αυτήν τη μελέτη, επικεντρωνόμαστε ειδικότερα στα μοτίβα σχεδιασμού παιχνιδιών για ατελείωτα mobile minigames, ένα σχετικά νέο είδος που έχει κερδίσει δημοτικότητα τα τελευταία χρόνια. Ωστόσο, λόγω της νεότητάς του, ο όρος "endless mobile minigames" δεν έχει εδραιωθεί διεθνώς και έχει λάβει περιορισμένη προσοχή στην έρευνα.

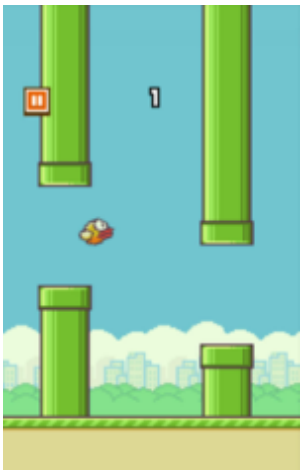
Ένα χαρακτηριστικό του είδους αυτού των παιχνιδιών είναι η γρήγορη διάρκεια τους, με κάθε επανάληψη συνήθως να διαρκεί από είκοσι δευτερόλεπτα έως λίγα λεπτά. Ο στόχος των παικτών είναι να σκοράρουν όσους περισσότερους πόντους μπορούν, μέχρι να συμβεί ένα συγκεκριμένο γεγονός που οδηγεί στην ήττα τους. Το ενδιαφέρον όσον αφορά τη σχεδίαση, είναι πως συνήθως δεν απαιτούν κανένα εγχειρίδιο ή οδηγίες για να ξεκινήσουν το παιχνίδι, λόγω της απλότητας τους. Έτσι, τα endless mobile minigames είναι ιδιαίτερα ελκυστικά για τους κατόχους κινητών τηλεφώνων, καθώς προσφέρουν σύντομες αλλά συναρπαστικές εμπειρίες και δυνατότητα κοινωνικής αλληλεπίδρασης με ανθρώπους από όλο τον κόσμο .

Μερικά από τα σημαντικά ατελείωτα παιχνίδια που έχουν κερδίσει δημοτικότητα είναι το Temple Run, το Doodle Jump και το Flappy Bird, με όλα τους να καταγράφουν πάνω από 10 εκατομμύρια λήψεις. Ωστόσο, ανάμεσα στα εκατομμύρια εφαρμογών που υπάρχουν στα καταστήματα εφαρμογών σήμερα, μόνο λίγες καταφέρνουν να φτάσουν στην κορυφή των δημοτικότητας / πληρωμένων / δωρεάν κατηγοριών, ενώ άλλες δεν έχουν καθόλου λήψεις.

Σημασία του όρου «Endless Mobile Games»

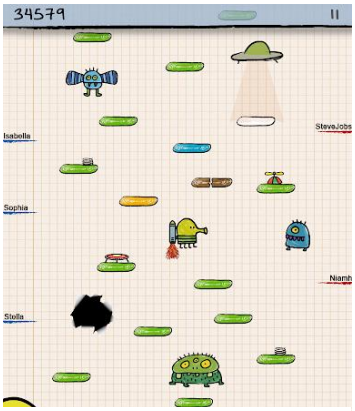
Η έννοια Endless Mobile Games βασίζεται στην ιδέα ότι ο παίκτης μπορεί θεωρητικά να σκοράρει πόντους απεριόριστα. Ο κύριος στόχος είναι να επιτευχθούν όσο το δυνατόν περισσότεροι πόντοι αντιμετωπίζοντας επανειλημμένα το σήμα λήξης του παιχνιδιού. Για παράδειγμα, στο παιχνίδι "Temple Run", ο παίκτης συνεχίζει να παίζει μέχρι ο χαρακτήρας του να συγκρουστεί με ένα εμπόδιο, με αποτέλεσμα το παιχνίδι να λήγει καθώς η γραμμή ζωής φθάνει στο μηδέν.

Τα Endless Mobile Games μπορούν να παρουσιαστούν με ποικίλους τρόπους, όπως Endless runners, endless flyers, endless avoidance και, γενικά, κάθε είδους αλληλεπίδραση που μπορεί να επαναλαμβάνεται απεριόριστα . Για να προσφέρουν μεγαλύτερη ποικιλία στο παιχνίδι, συνήθης προσέγγιση είναι η χρήση ενός τυχαίου αλγορίθμου για τη δημιουργία εμποδίων σε διάφορα επίπεδα , τα οποία πρέπει να ξεπεράσει ο παίκτης. Ένα κρίσιμο στοιχείο εδώ είναι ότι η δυσκολία του παιχνιδιού αυξάνεται σταδιακά σε σύντομο χρονικό διάστημα. Η ευκολία των οθονών αφής των smartphones έχει απλοποιήσει τους ελέγχους των κινητών παιχνιδιών . Πράγματι, πολλά ατελείωτα παιχνίδια χρησιμοποιούν μόνο ένα και μοναδικό πάτημα, που ελέγχει ολόκληρο το gameplay, όπως στο παιχνίδι "Flappy Bird".



Flappy Bird

Temple Run



Doodle Jump

ΜΕΡΟΣ ΔΕΥΤΕΡΟ

ΥΛΟΠΟΙΗΣΗ

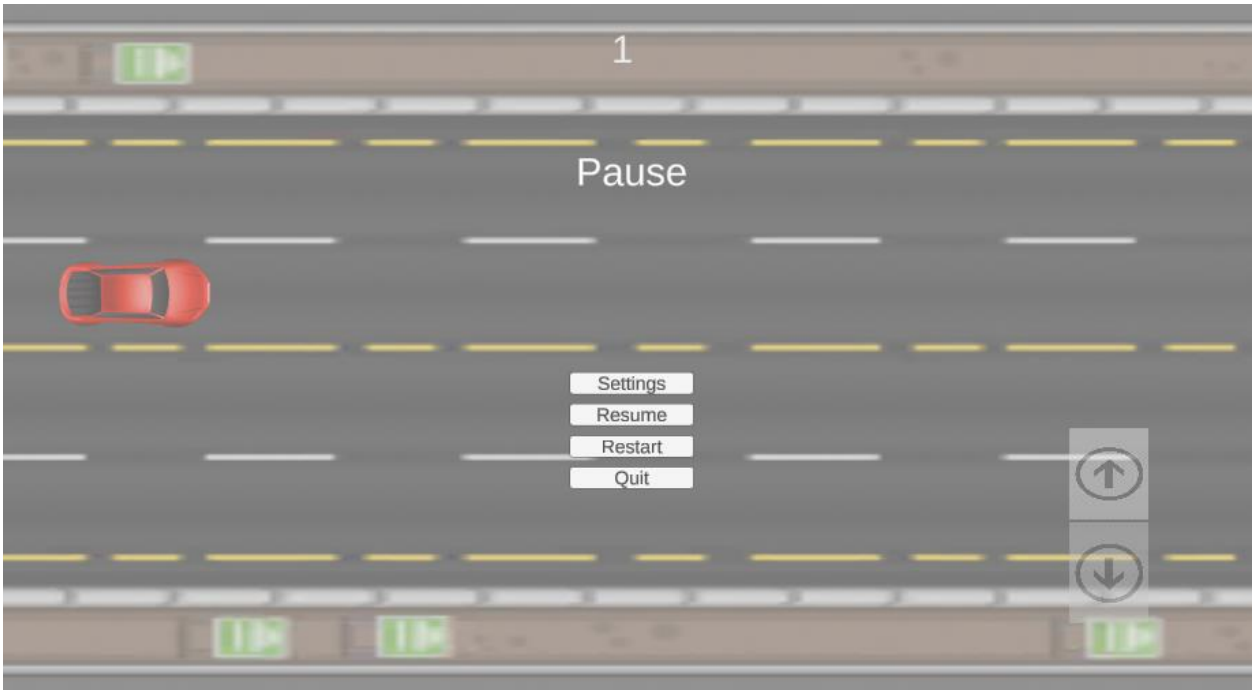
Εκκίνηση παιχνιδιού

Το παιχνίδι ξεκινάει από την αρχική οθόνη στην οποία το αυτοκίνητο του παίκτη φαίνεται να κινείται σε έναν δρόμο χωρίς εμπόδια, ενώ στο κάτω μέρος της οθόνης βρίσκεται το παράθυρο οπού πρέπει να καταγράψει το όνομα του. Έτσι το παιχνίδι ψάχνοντας την βάση δεδομένων βρίσκει τον παίκτη και εμφανίζει το κουμπί START. Στην περίπτωση που το όνομα χρήστη δεν βρεθεί ο παίκτης θα αναγκαστεί να δημιουργήσει έναν νέο χρήστη πατώντας το κουμπί «δημιουργία χρήστη» που βρίσκεται στο κάτω δεξιά μέρος της οθόνης και υστέρα να συνεχίσει.

Με το πάτημα του START η αρχική οθόνη εξαφανίζεται και αρχίζουν να εμφανίζονται άλλα αυτοκίνητα στην οθόνη τα οποία λειτουργούν ως εμπόδια και δυο κουμπιά σε σχήμα βέλους. Με το πάτημα των κουμπιών ο παίκτης κινεί το αυτοκίνητο του στον οριζόντιο άξονα, έτσι ώστε να αποφύγει τα εμπόδια.

Ακόμη, εμφανίζεται το κουμπί «Pause».

Το οποίο φέρνει το παιχνίδι σε κατάσταση παύσης. Στην κατάσταση αυτή, ο χρήστης έχει την δυνατότητα να ξεκινήσει το παιχνίδι από την αρχή, χωρίς όμως να υποθηκευθεί τον σκορ που έχει πέτυχει μέχρι εκείνη τη στιγμή, να πραγματοποιήσει έξοδο από την εφαρμογή, να συνεχίσει το παιχνίδι και να αποκτήσει πρόσβαση στο παράθυρο ρυθμίσεις (settings).

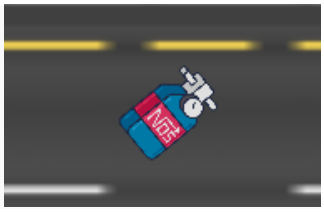


Στο παράθυρο αυτό μπορεί μέσω των κυλιόμενων μπαρών που φαίνονται παρακάτω να ρυθμίσει την φωτεινότητα και τον ήχο της εφαρμογής.


0
Settings
Sound
Brightness
Cancel

Power ups

Κατά την διάρκεια του παιχνιδιού, ο παίκτης μπορεί να πάρει κάποια power ups τα οποία τον βοηθούν να πέτυχει μεγαλύτερο σκορ. Το ένα power up δίνει στο παίκτη περισσότερη ταχύτητα για 5 δευτερόλεπτα και κάνει το μοντέλο του αυτοκίνητου να βγάζει φωτιές από το πίσω μέρος κατά τη διάρκεια του.



Το δεύτερο power up αφαιρεί την σύγκρουση με άλλα αυτοκίνητα για μερικά δευτερόλεπτα και κάνει το μοντέλο του αυτοκίνητου να αλλάζει χρώματα με γρήγορο ρυθμό κατά τη διάρκεια του.

Το τρίτο power up διπλασιάζει τους πόντους που παίρνει ο παίκτης για 3 δευτερόλεπτα από τη στιγμή που θα το χρησιμοποιήσει.

Game Over

Στην περίπτωση που ο παίκτης χτυπήσει σε κάποιο από τα εμπόδια το παιχνίδι τελειώνει και εμφανίζεται η τελική οθόνη στην οποία ο παίκτης έχει την δυνατότητα να ξανακινήσει ή να βγει εντελώς από την

εφαρμογή. Στην οθονη αυτή, ακομη φαινεται ο πινακας score στον οποιο ο κάθε παικτης μπορει να δει τις επιδόσεις του όπως και λολων των υοποιοπων παικτων που εχουν λογαριασμο στην εφαρμογη.

Βάση Δεδομένων

Google Firebase

Για την υλοποίηση των online χαρακτηριστικών της η εφαρμογή χρησιμοποιεί μια υπηρεσία της google με όνομα Google firebase. Το Google Firebase είναι μια ολοκληρωμένη πλατφόρμα που παρέχεται από την Google για την ανάπτυξη εφαρμογών διαδικτύου και κινητών συσκευών. Προσφέρει μια ευρεία γκάμα

υπηρεσιών και εργαλείων για να βοηθήσει τους προγραμματιστές να χτίσουν, βελτιώσουν και διαχειριστούν τις εφαρμογές τους με μεγαλύτερη αποτελεσματικότητα. Το Firebase περιλαμβάνει διάφορες λειτουργίες, όπως:

1. Πραγματικού χρόνου βάση δεδομένων: Μια NoSQL βάση δεδομένων στον νέφος που επιτρέπει πραγματικού χρόνου συγχρονισμό και αποθήκευση δεδομένων σε πολλούς πελάτες.

2. Πιστοποίηση: Μια υπηρεσία που διαχειρίζεται την πιστοποίηση χρηστών και επιτρέπει στους χρήστες να συνδεθούν χρησιμοποιώντας διάφορες μεθόδους, όπως ηλεκτρονικό ταχυδρομείο, Google, Facebook κ.λπ.

3. Cloud Firestore: Άλλη μια βάση δεδομένων NoSQL στον νέφος με πιο προηγμένες λειτουργίες ερωτήσεων και κλιματιζόμενοτητα από την πραγματικού χρόνου βάση δεδομένων.

4. Cloud Functions: Ανυπόληπτες λειτουργίες που επιτρέπουν στους προγραμματιστές να εκτελούν προσαρμοσμένο κώδικα ανταποκρινόμενοι σε γεγονότα που προκαλούνται από το Firebase ή άλλες υπηρεσίες.

5. Cloud Storage: Μια λύση αποθήκευσης στο νέφος για την αποθήκευση και παροχή περιεχομένου που δημιουργείται από τον χρήστη, όπως εικόνες, βίντεο και άλλα αρχεία.

6. Φιλοξενία: Η Firebase Hosting επιτρέπει γρήγορη και ασφαλή φιλοξενία για web apps με ενσωματωμένα πιστοποιητικά SSL και υποστήριξη δικτύου παράδοσης περιεχομένου (CDN).

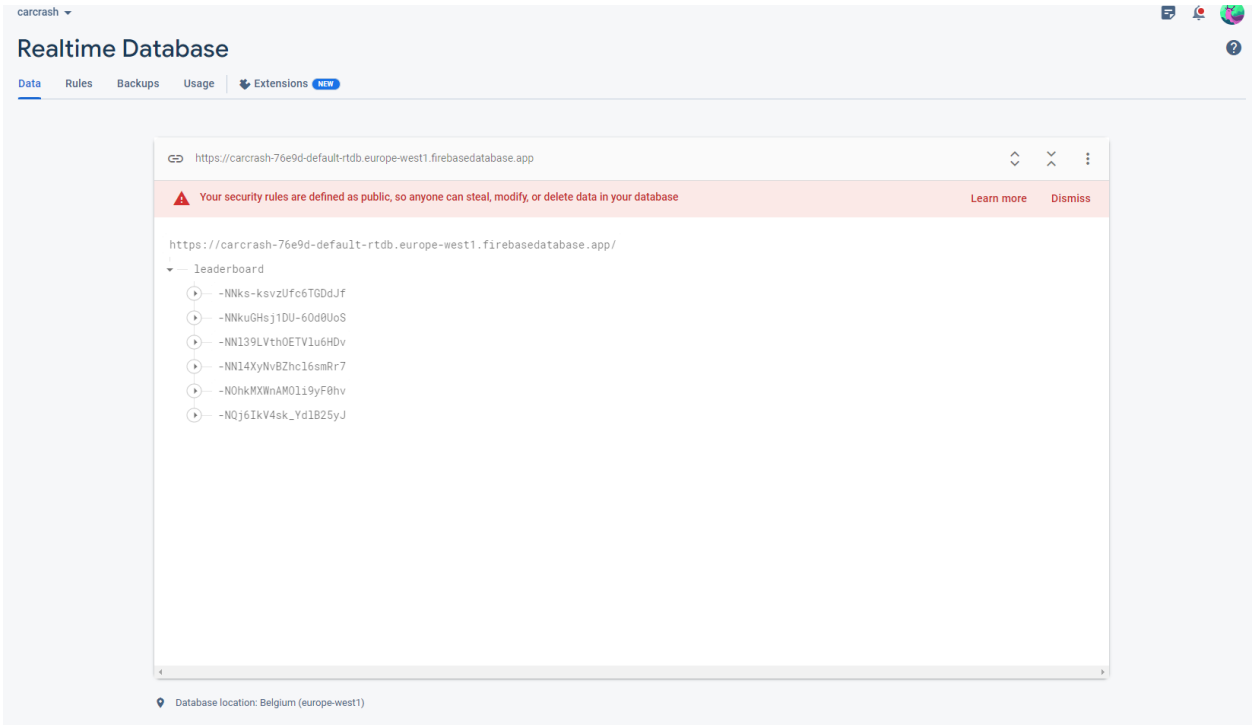
7. Cloud Messaging: Μια υπηρεσία που διευκολύνει την αποστολή ειδοποιήσεων και μηνυμάτων στους χρήστες σε διάφορες πλατφόρμες.

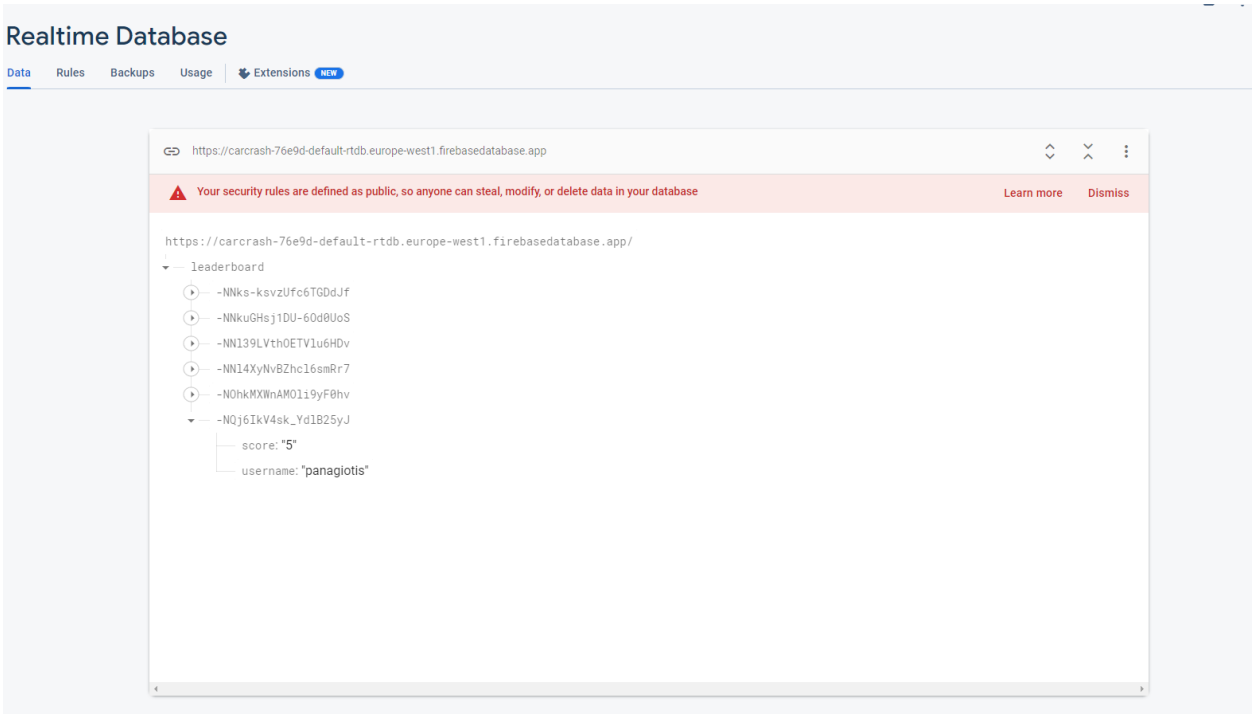
8. Παρακολούθηση Απόδοσης: Εργαλεία για την παρακολούθηση και ανάλυση της απόδοσης της εφαρμογής σας, επιτρέποντάς σας να εντοπίζετε και επιδιορθώνετε προβλήματα με ταχύτητα.

9. Test Lab: Μια υπηρεσία για τον δοκιμή της εφαρμογής σας σε πραγματικές συσκευές στο νέφος, επιτρέποντάς σας να αξιολογήσετε την απόδοσή και τη

Υλοποίηση

Η firebase λειτουργεί ως βάση δεδομένων της εφαρμογής και μέσα της αποθηκεύονται οι χρήστες ξεκινώντας με μηδέν πόντους σκορ. Μετά το τέλος του παιχνιδιού η βάση ενημερώνεται με τα τελευταία σκορ μεγαλύτερα από αυτό που υπήρχαν μέχρι εκείνη τη στιγμή αποθηκευμένα στη βάση. Στην αντίθετη περίπτωση κάτω από το όνομα του χρήστη θα μείνει το σκορ που είναι ήδη γραμμένο.




Realtime Database
Data
Rules
Backups
Usage
Extensions
NEW
https://carcrash-76e9d-default-rtdb.europe-west1.firebasedatabase.app
Your security rules are defined as public, so anyone can steal, modify, or delete data in your database
Learn more
Dismiss
https://carcrash-76e9d-default-rtdb.europe-west1.firebasedatabase.app/
leaderboard
-NNks-ksvzUfc6TGDdJf
-NNkuGHsj1DU-6Od0UoS
-NN139LVthOETV1u6HDv
-NN14XyNvBZhcl6smRr7
-NOhkMXWnAMOli9yF0hv
-NQj6IkV4sk_YdlB25yJ
score: "5"
username: "panagiotis"

Scripts Εφαρμογής

Η λειτουργία της εφαρμογής βασίζεται στα παρακάτω scripts :

PLayer.cs	Το script αυτό ελέγχει όλες τις λειτουργίες του παίκτη, θα αναλυθεί περισσότερο στη συνέχεια.
Score.cs	Χρησιμοποιείται για να μετράει το score και να το δείχνει στο αντίστοιχο panel κατά τη διάρκεια του παιχνιδιού.
LoopingBackground.cs	Χρησιμοποιείται για να προχωράει το background, για να φαίνεται ότι ο παίκτης κινείται .
spawnObstacles.cs	Χρησιμοποιείται για να εμφανίζει τα εμπόδια και τα powerups, θα αναλυθεί περισσότερο στη συνέχεια.
Obstacle.cs	Ελέγχει τις ενέργειες που κάνουν τα εμπόδια, δηλαδή όταν κάνει collide με τον παίκτη καταστρέφει το object του παίκτη και τελειώνει το παιχνίδι, ενώ όταν κάνει collide με κάποιο border, δηλαδή βγαίνει έξω από την κάμερα καταστρέφεται το ίδιο για να εξοικονομηθεί μνήμη.
speedUP.cs	Ελέγχει τις ενέργειες που κάνει το speed up power up, δηλαδή να αυξάνει το gametime αν 2 για ορισμένο χρόνο, με αποτέλεσμα να φαίνεται ότι το αυτοκίνητο τρέχει πιο γρηγορά ,ενώ ταυτόχρονα παίζει το nitro animation.
doublePOINTS.cs	Ελέγχει τις ενέργειες που κάνει το double points power up, δηλαδή να αυξάνει ανά δυο τους πόντους του παίκτη για ορισμένο χρόνο.

BackgroundMusic.cs	παίζει μουσική κατά τη διάρκεια του παιχνιδιού και σταματάει όταν το παιχνίδι τελειώνει.
CameraMovement.cs	Χρησιμοποιείται για να να προχωράει την κάμερα, για να φαίνεται ότι ο παίκτης κινείται .
StartGame.cs	Το script χρησιμοποιείται για να επιτρέψει στον παίκτη να να φτιάξει έναν χρήστη και να συνδεθεί στο παιχνίδι, ενεργοποιήσει τα spawners που εμφανίζουν τα εμπόδια και τα power ups, την καταγραφή score και τη δυνατότητα του χρήστη να μετακινεί το αυτοκίνητο με τα βελάκια, ενώ κλείνει το αρχικό panel. Χρησιμοποιείται από το κουμπί START.
GameOver.cs	Το script χρησιμοποιείται για να απενεργοποιήσει τα spawners,την καταγραφή score και τη δυνατότητα του χρήστη να μετακινεί το αυτοκίνητο με τα βελάκια, ενώ καταγραφεί το υπάρχον σκορ στη βάση δεδομένων και ανοίγει το τελικό panel . Καλείται όταν γίνεται destroyed το αντικείμενο player.
User.cs	Το script αυτό περιέχει την Class που χρησιμοποιείται για την δημιουργία των user objects με τα οποία γίνονται οι εγγραφές στην βάση
Brightness.cs	Το script αυτό χρησιμοποιείται για να αλλάξει τιμή στην φωτεινότητα της εφαρμογής.

Σημαντικότερα Scripts

Player.cs

Το PLayer.cs είναι το script στο οποίο βασίζεται η κύρια λειτουργία της εφαρμογή. Μέσω αυτό ο χρήστης μπορεί να μετακινεί το αυτοκίνητο στον άξονα Y. επίσης ελέγχει τις λειτουργίες των power ups που θα παίρνει όπως και τον χρόνο διάρκειας τους με τη χρήση ενός διαφορετικού timer για το καθένα.

PLayer.cs GameOver.cs

Miscellaneous Files PLayer

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

public class PLayer : MonoBehaviour
{
    public float playerSpeed;
    private Rigidbody2D rb;
    private Vector2 playerDirection;
    public bool doublePOints=false;
    public float timeRemainingDP = 3;
    public float timeRemainingSU = 3;
    private bool startTimerDP = false;
    private bool startTimerSU = false;
    public Animator animator;

    // Start is called before the first frame update
    void Start()
    {
        rb = GetComponent<Rigidbody2D>();
    }

    // Update is called once per frame
    void Update()
    {

        if (StartGame.started)
        {
            float directionY = Input.GetAxisRaw("Vertical");
            playerDirection = new Vector2(0, directionY).normalized;
        }


        if (startTimerDP)
        {

            if (timeRemainingDP > 0)
            {
                timeRemainingDP -= Time.deltaTime;
            }
            else
            {
                startTimerDP = false;
                doublePOints = false;

                timeRemainingDP = 3;



            }
        }
```

```
52          if (startTimerSU)
53          {
54
55              if (timeRemainingDP > 0)
56              {
57                  timeRemainingDP -= Time.deltaTime;
58              }
59              else
60              {
61                  startTimerSU = false;
62
63                  animator.SetBool("nitro", false);
64                  Time.timeScale = 1;
65                  timeRemainingDP = 10;
66
67
68              }
69          }
70      }
71      void FixedUpdate()
72      {
73          rb.velocity = new Vector2(0, playerDirection.y * playerSpeed);
74      }
75
76      private void OnTriggerEnter2D(Collider2D collision)
77      {
78
79
80          if (collision.tag == "DoublePoints")
81          {
82              animator.SetBool("nitro", true);
83              doublePOints = true;
84              startTimerDP = true;
85
86          }
87
88          if (collision.tag == "SpeedUP")
89          {
90              Debug.Log("aaaaaaa");
91              animator.SetBool("nitro", true);
92              Time.timeScale = 2;
93              startTimerSU = true;
94          }
95      }
96  }
97
```

spawnObstacles.cs

Το spawnObstacles.cs χρησιμοποιείται για να δημιουργήσει spawners τα οποία μπορούν να εμφανίζουν ένα random από ένα δοσμένο σύνολο, ανά χρονικό διάστημα που έχει επιλεγεί από τον προγραμματιστή. Στην προκειμένη περίπτωση χρησιμοποιείται για να εμφανίζει τα εμπόδια, τα οποία έρχονται από ένα σύνολο τεσσάρων διαφορετικών αυτοκινήτων, αλλά και power ups τα οποία έρχονται από ένα σύνολο 2 διαφορετικών. Περιέχει την συνάρτηση Spawn(), η οποία εμφανίζει τα prefabs των εμποδίων και των

powerups στην οθόνη σε σημεία με συντεταγμένες που παίρνουν διαφορετική τυχαία τιμή κάθε φορά που καλούνται.

```
public class spawnObstacles : MonoBehaviour
{
    public GameObject obstacle1;
    public GameObject obstacle2;
    public GameObject obstacle3;
    public GameObject obstacle4;
    public float maxX;
    public float minX;
    public float maxY;
    public float minY;
    public float timeBetweenSpawn;
    private float spawnTime;

    // Update is called once per frame
    void Update()
    {
        if (GameObject.FindGameObjectWithTag("Player") != null)
        {



            if (Time.time > spawnTime)
            {
                Spawn();
                spawnTime = Time.time + timeBetweenSpawn;
            }
        }
    }

    void Spawn()
    {
        float randomX = Random.Range(minX, maxX);
        float randomY = Random.Range(minY, maxY);


        int randomValue = Random.Range(1, 5);
        switch (randomValue)
        {
            case 1:
            Instantiate(obstacle1, transform.position + new Vector3(randomX, randomY, 0), transform.rotation);
                break;
            case 2:
            Instantiate(obstacle2, transform.position + new Vector3(randomX, randomY, 0), transform.rotation);
                break;
            case 3:
            Instantiate(obstacle3, transform.position + new Vector3(randomX, randomY, 0), transform.rotation);
                break;
            case 4:
            Instantiate(obstacle4, transform.position + new Vector3(randomX, randomY, 0), transform.rotation);
                break;


        }
    }
}
```

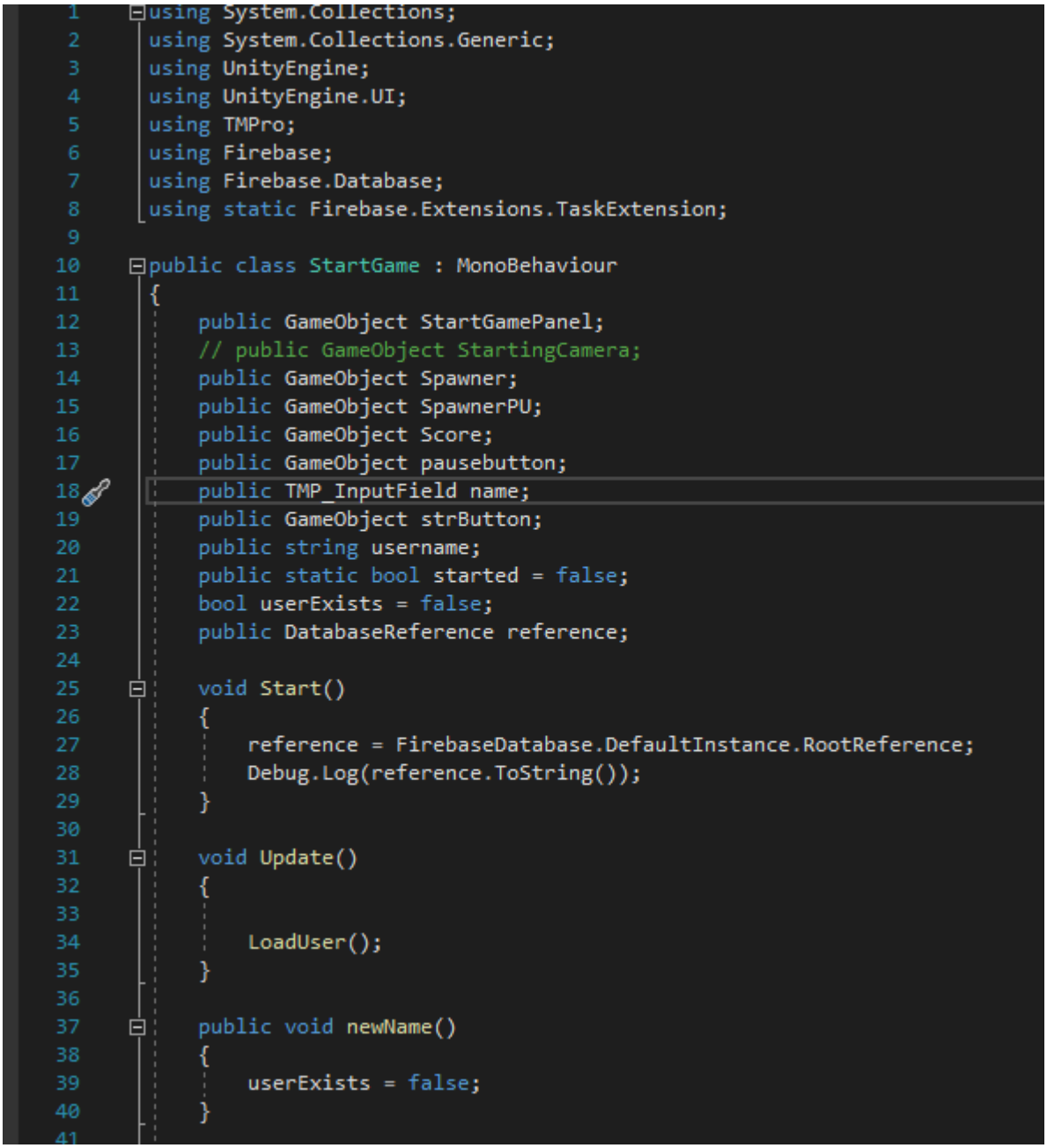
StartGame.cs

ΤΟ StartGame.cs χρησιμοποιείται για να ξεκινήσει το παιχνίδι αλλά και για να αποθηκεύσει τον χρήστη στη βάση δεδομένων , ή αντίστοιχα να τον βρει αν ήδη υπάρχει. Κατά την εκκίνηση του script δημιουργείται ένα reference στην βάση δεδομένων firebase. Το script περιέχει την συνάρτηση GameStart(), η οποία εξαφανίζει από την οθόνη το αρχικό πάνελ και ενεργοποιεί τα spawner των εμποδίων και των powerups. Επίσης ξεκινάει να μετράει σκορ και δηλώνει το παιχνίδι ως «started» στην ομώνυμη μεταβλητή. Ακόμη, το script περιέχει τις συναρτήσεις LoadUser(), CreateUser(), searchUser() και newname().

Η newName() αλλαζει την μεταβλητη userExists σε false. Η searchUser() δημιούργει έναν πίνακα userindb και ένα reference στο child της βάσης με όνομα “leaderboard” από το οποίο παίρνει όλα τα values μέσω ενός thread. Το task του thread είναι να πάρει το snapshot της βάσης και να το μετατρέψει σε πίνακα του οποίου το κάθε κελί θα αντιστοιχεί σε μια εγγραφή της βάσης. Υστέρα για κάθε εγγραφή δημιουργείται ένα νέο reference το οποίο βλέπει τα περιεχόμενα της και δημιουργεί ένα object τύπου “User” για το καθένα από αυτά. Η class User έχει username και score για το κάθε object της, τα οποία παίρνει από τον πίνακα. Αν το username βρεθεί σε κάποια από τις εγγραφές του πίνακα τότε η μεταβλητή userExists γίνεται αληθής και η επανάληψη σταματάει.

Η LoadUser() ελέγχει αν ο χρήστης έχει πληκτρολογήσει κάποιο όνομα στο παράθυρο. Αν το παράθυρο δεν είναι κενό καλεί την searchUser() με παράμετρο το όνομα που υπάρχει στο παράθυρο αν μετά την κλήση η μεταβλητή userExist είναι αληθής τότε εμφανίζει το κουμπί “start” .

Τέλος, η συνάρτηση CreateUser() καλείται αν πατηθεί το κουμπί “create new user” και διαβάζει το όνομα που έχει γράψει ο χρήστης στο παράθυρο. Αν το όνομα δεν είναι κενό τότε καλεί την συνάρτηση searchUser(). Στην περίπτωση που ο χρήστης δεν υπάρχει ήδη δημιουργεί ένα νέο αντικείμενο τύπου User το οποίο έχει ως username το όνομα που έχει πληκτρολογήσει ο χρήστης και ως σκορ τον αριθμό «0» και το αποθηκεύει στην βάση της firebase.



```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using TMPro;
using Firebase;
using Firebase.Database;
using static Firebase.Extensions.TaskExtension;

public class StartGame : MonoBehaviour
{
    public GameObject StartGamePanel;
    // public GameObject StartingCamera;
    public GameObject Spawner;
    public GameObject SpawnerPU;
    public GameObject Score;
    public GameObject pausebutton;
    public TMP_InputField name;
    public GameObject strButton;
    public string username;
    public static bool started = false;
    bool userExists = false;
    public DatabaseReference reference;

    void Start()
    {
        reference = FirebaseDatabase.DefaultInstance.RootReference;
        Debug.Log(reference.ToString());
    }

    void Update()
    {

        LoadUser();
    }

    public void newName()
    {
        userExists = false;
    }

```

```
41
42      public void LoadUser()
43      {
44          if (name.text != "")
45          {
46              searchUser(name.text);
47
48              Debug.Log(userExists);
49              if (userExists)
50              {
51                  username = name.text;
52                  strButton.SetActive(true);
53              }
54              else
55              {
56                  Debug.Log("username not exists");
57                  strButton.SetActive(false);
58
59              }
60          }
61
62          else
63              Debug.Log("no username");
64
65      }
66
67
68
69      public void GameStart()
70      {
71          pausebutton.SetActive(true);
72          StartGamePanel.SetActive(false);
73          Spawner.SetActive(true);
74          SpawnerPU.SetActive(true);
75          Score.SetActive(true);
76          started = true;
77      }
78
```

```
77      }
78
79      public void CreateUser()
80      {
81          username = name.text;
82          if (username != "")
83          {
84              searchUser(username);
85              if (!userExists)
86              {
87                  User user = new User(username, "0");
88                  string json = JsonUtility.ToJson(user);
89                  Debug.Log(reference.ToString());
90                  reference.Child("leaderboard").Push().SetRawJsonValueAsync(json);
91                  Debug.Log("user created");
92              }
93              else
94              {
95                  Debug.Log("username already exists");
96              }
97          }
98          else
99              Debug.Log("No username..");
100
101
102
103     }
104
```

```
105     public void searchUser(string username)
106     {
107         string[] userindb = new string[100];
108         FirebaseDatabase.DefaultInstance.RootReference.Child("leaderboard").GetValueAsync()
109         .ContinueWithOnMainThread(task => {
110             if (task.IsFaulted)
111             {
112 
113                 Debug.Log("error...");
114             }
115             else if (task.IsCompleted)
116             {
117 
118                 int i = 0;
119                 DataSnapshot snapshot = task.Result;
120                 foreach (var childSnapshot in snapshot.Children)
121                 {
122 
123                     userindb[i] = childSnapshot.Key.ToString();
124                     // Debug.Log(userindb[i]);
125                     i++;
126                 }
127                 // Debug.Log(string.Join(",", userindb));
128 
129                 for (int j = 0; j < 100; j++)
130                 {
131 
132                     FirebaseDatabase.DefaultInstance.RootReference.Child("leaderboard").Child(userindb[j]).GetValueAsync()
133                     .ContinueWithOnMainThread(
134                      task => {
135                         if (task.IsFaulted)
136                         {
137 
138                             Debug.Log("error...");
139                         }
140                         else if (task.IsCompleted)
141                         {
142 
143                             DataSnapshot snapshot = task.Result;
144 
145                             User user = JsonUtility.FromJson<User>(snapshot.GetRawJsonValue());
146                             if (username == user.username.ToString())
147                             {
148 
149                                 userExists = true;
150                                 break;
151 
152                             }
153 
154 
155 
156 
157                         }
158                         Debug.Log(userExists);
159 
```

GameOver.cs

ΤΟ GameOver.cs χρησιμοποιείται για να τερματίσει το παιχνίδι αλλά και για να αποθηκεύσει το σκορ του εκάστοτε χρήστη στη βάση δεδομένων. Κατά την εκκίνηση του script δημιουργείται ένα reference στην βάση δεδομένων firebase. Το script περιέχει τις συναρτήσεις Quit() και restart(), οι οποίες σβήνουν ή επανεκκινούν την εφαρμογή αντίστοιχα. Ακόμη, το script περιέχει τις συναρτήσεις Downloadlb(), getScore(), uploadScore().

Η συνάρτηση uploadScore() λαμβάνει ως παράμετρο μια μεταβλητή τύπου String και δημιουργεί ένα αντικείμενο τύπου “User” με username και score τα δεδομένα που έχει πάρει από το τις ομώνυμες

μεταβλητές. Μετατρέπει έτσι το User στην μεταβλητή jason τύπου String η οποία μπορεί να διαβαστεί από τη βάση και την ανεβάζει στο child της βάσης με όνομα την μεταβλητή που έχει λάβει ως παράμετρο .

Η συνάρτηση getScore() λαμβάνει ως παράμετρο μια μεταβλητή τύπου String και δημιούργει ένα reference στο child της βάσης με όνομα το περιεχόμενο της μεταβλητής που έχει λάβει ως παράμετρο από το οποίο παίρνει όλα τα values μέσω ενός thread. Το task του thread είναι να πάρει το snapshot της βάσης και να το μετατρέψει σε αντικείμενο τύπου User. Έπειτα πρέπει να ελέγξει αν το περιεχόμενο της μεταβλητής username είναι το ίδιο με το username του νέου αντικειμένου και το περιεχόμενο της μεταβλητής score μικρότερο από το score του νέου αντικειμένου. Αν και τα δυο αληθεύουν τότε αποθηκεύει το νέο σκορ στη βάση, στη θέση του προηγουμένου, καλώντας την συνάρτηση uploadScore(), αλλιώς το νέο σκορ διαγράφεται και στη βάση μένει το προηγούμενο σκορ του χρήστη.

Η συνάρτηση Downloadlb() δημιούργει έναν πίνακα userindb και ένα reference στο child της βάσης με όνομα “leaderboard” από το οποίο παίρνει όλα τα values μέσω ενός thread. Το task του thread είναι να πάρει το snapshot της βάσης και να το μετατρέψει σε πίνακα του οποίου το κάθε κελί θα αντιστοιχεί σε μια εγγραφή της βάσης. Ύστερα, καλεί τη συνάρτηση getScore() για κάθε κελί του πίνακα. Σε κάθε frame της εφαρμογής, κατά το update το script αυτό κρατάει αποθηκευμένο το όνομα του χρήστη όπως και το σκορ που έχει μέχρι εκείνη τη χρονική στιγμή.

Όταν το αντικείμενο “Player” καταστραφεί κατά την σύγκρουση του με κάποιο από τα εμπόδια ενεργοποιείται το πάνελ τερματίσου παιχνιδιού και καλείται η συνάρτηση Downloadlb(). Μόλις κληθεί η συνάρτηση η μεταβλητή “hasrun” γίνεται αληθής έτσι ώστε να σταματήσει την Downloadlb() από το να κληθεί εκ νέου στο επόμενο update.

StartGame.cs spawnObstacles.cs GameOver.cs*

Miscellaneous Files GameOver

```
1   using System.Collections;
2   using System.Collections.Generic;
3   using System;
4   using UnityEngine;
5   using UnityEngine.SceneManagement;
6   using UnityEngine.UI;
7   using Firebase;
8   using Firebase.Database;
9   using static Firebase.Extensions.TaskExtension;
10  using TMPro;
11
12  public class GameOver : MonoBehaviour
13  {
14      public GameObject gameOverPanel;
15      public GameObject pausebutton;
16      public DatabaseReference reference;
17      public string username;
18      public string score;
19      public GameObject strpanel;
20      public TMP_Text leaderboard;
21      public GameObject scorelabel;
22      public bool hasrun = false;
23
24
25
26
27      void Start()
28      {
29          reference = FirebaseDatabase.DefaultInstance.RootReference;
30
31      }
32
33      void uploadScore(string currentuser)
34      {
35          User user = new User(username, score);
36          string json = JsonUtility.ToJson(user);
37
38          reference.Child("leaderboard").Child(currentuser).SetRawJsonValueAsync(json);
39          Debug.Log("egine"+ json + currentuser);
40      }
```

```
    void Update()
    {

        username = strpanel.GetComponent<StartGame>().username;
        score = ((int)scorelabel.GetComponent<Score>().score).ToString();

        if (GameObject.FindGameObjectWithTag("Player") == null)
        {
            StartGame.started = false;

            gameOverPanel.SetActive(true);
            if (!hasrun)
            {
                //uploadScore();
                Downloadlb();


                hasrun = true;
            }



            if (Input.GetKeyDown(KeyCode.Escape))
            {
                Quit();
            }

            if (Input.GetKeyDown(KeyCode.Return))
            {
                Restart();
            }
        }


    }

    public void Restart()
    {
        SceneManager.LoadScene(SceneManager.GetActiveScene().name);
        pausebutton.SetActive(false);
        Time.timeScale = 1;

    }

    public void Quit()
    {

            Application.Quit();

    }
```

```
91      }
92
93      public void Downloadlb()
94      {
95          //string[] lb = new string[100];
96          string[] userindb=new string[100];
97          FirebaseDatabase.DefaultInstance.RootReference.Child("leaderboard").GetValueAsync()
98              .ContinueWithOnMainThread(task => {
99                  if (task.IsFaulted)
100                 {
101
102                     Debug.Log("error...");
103                 }
104                 else if (task.IsCompleted)
105                 {
106
107                     int i = 0;
108                     DataSnapshot snapshot = task.Result;
109                     foreach (var childSnapshot in snapshot.Children)
110                     {
111
112                         userindb[i] = childSnapshot.Key.ToString();
113                         Debug.Log(userindb[i]);
114                         i++;
115                     }
116                     // Debug.Log(string.Join(",", userindb));
117
118                     for (int j = 0; j < 100; j++)
119                     {
120                         getScore(userindb[j]);
121
122                     }
123
124                 }
125             });
126
127     }
```

```
128
129     public void getScore(string currentuser)
130     {
131         string userstatues="";
132         FirebaseDatabase.DefaultInstance.RootReference.Child("leaderboard").Child(currentuser).GetValueAsync()
133             .ContinueWithOnMainThread(
134             task => {
135                 if (task.IsFaulted)
136                 {
137
138                     Debug.Log("error...");
139                 }
140                 else if (task.IsCompleted)
141                 {
142
143                     DataSnapshot snapshot = task.Result;
144
145                     User user = JsonUtility.FromJson<User>(snapshot.GetRawJsonValue());
146                     if (user.username.ToString() == username && Int32.Parse(user.score) < Int32.Parse(score))
147                     {
148
149                         uploadScore(currentuser);
150                         userstatues = user.username.ToString() + " " + score.ToString();
151
152                     }
153                     else
154                     {
155                         userstatues = user.username.ToString() + " " + user.score.ToString();
156                     }
157                     leaderboard.text = userstatues + "\n" + leaderboard.text;
158                     //Debug.Log(userstatues);
159
160                 }
161             });
162
163     }
164
165
```

Πηγές

Βιβλιογραφία

- Sun, Y., Zhao, Y., Jia, S., Zheng, D., Understanding the Antecedents of Mobile Game Addiction: The Roles of perceived Visibility perceived Enjoyment and Flow, National Natural Science Foundation of China, Project No. 71201118, Hubei Province Science and Technology Support Program No. 2014BDF106, 2014.
- Vasconcelos de Medeiros, R.J., Vasconcelos de Medeiros, T.F., Procedural Level Balancing in Runner Games, SBC - Proceedings of the SBGames, ISSN: 2179- 2259, 2014.
- Park, Hyun Jung and Kim, Sang-hoon, A Bayesian network approach to examining key success factors of mobile games, Publisher: Elsevier Inc., issn 0148-2963, doi 10.1016/j.jbusres.2012.02.036, pp. 13531359, 2013.
- Oh, Chang-su Kim Eun-hai and Hoon, Kyung and Kim, Jae Kyung, The appealing characteristics of download type mobile games, doi 10.1007/s11628-009-0088-0, pp. 253-269, 2010.
- Rubem Jose Vasconcelos de Medeiros, Tacio Filipe Vasconcelos de Medeiros, Procedural Level Balancing in Runner Games, ISSN: 2179-2259, pp. 797-801, 2014.

Assets

- Car_Sprites: https://opengameart.org/content/free-top-down-car-sprites-by-unlucky-studio
- Background: https://opengameart.org/content/2d-top-down-highway-background
- Music: https://www.youtube.com/watch?v=ypvFlsOetOA&ab_channel=Machinimasound