ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

“Ανάπτυξη-πολυχρηστικού-παιχνιδιού-με-Javascript-και-Canvas”

ΟΝΟΜΑΤΕΠΩΝΥΜΟ ΦΟΙΤΗΤΗ: Λάμπρος Δίπλας, ΑΜ:1906

ΕΠΙΒΛΕΠΩΝ:Σωτήρης Χριστοδούλου

ΑΝΤΙΡΡΙΟ 2019

Εγκρίθηκεαπότην τριμελήεξεταστικήεπιτροπή

Αντίρριο, 2019,

ΕΠΙΤΡΟΠΗΑΞΙΟΛΟΓΗΣΗΣ

1. ,Υπογραφή

2. ,Υπογραφή

3. ,Υπογραφή

Περίληψη

Σκοπός της πτυχιακής εργασίας είναι να δημιουργήσουμε ένα παιχνίδι στον υπολογιστή. Η δημιουργία ενός ShootingGame δηλαδή ένα παιχνίδι στο οποίο θα πυροβολούμε εικονικά κάποια αντικείμενα. Στο παιχνίδι μας τα αντικείμενά μας θα είναι δυο ιπτάμενα πουλιά τα οποία όσο περνάει ο χρόνος μέσα στο παιχνίδι θα μας δυσκολεύουν όλο και περισσότερο με τις κινήσεις τους. Αρχικά για την επίτευξη του θα χρειαστεί να χρησιμοποιήσουμε σε μεγάλο βαθμό τρία βασικά προγράμματα το Krita, Audacity και το Visual Studio Code καθώς και δυο από τις ποιο σημαντικές γλώσσες προγραμματισμού την JavaScript (JS) και την HyperText Markup Language(HTML). Τέλος θα δούμε πως συνεργάζονται και λειτουργούν μαζί όλα αυτά έτσι ώστε να μπορούμε να παίξουμε το παιχνίδι μας με επιτυχία.

Πίνακας περιεχομένων

Κεφάλαιο 1 – Εισαγωγή

1.1 Το παιχνίδι

Όταν ο παίκτης θα πυροβολεί δυο ιπτάμενά πουλιά τότε θα εμφανίζεται ένας στόχος στο background μας. Πάνω δεξιά στην οθόνη δηλαδή στο background θα υπάρχει ένα χρονόμετρο που θα σταματάει στο 1 λεπτό και το παιχνίδι θα τελειώνει. Πάνω αριστερά θα υπάρχει ένα πλαίσιο για το σκορ του παιχνιδιού όπου θα παίρνει +100 πόντους όταν πετύχει το ιπτάμενο πουλί ενώ κάθε φορά που αστοχεί χάνει -10 πόντους. Είτε ο παίκτης πετύχει το στόχο είτε αστοχήσει τα ιπτάμενα πουλιά αρχίζουν να κουνιούνται όλο και πιο γρήγορα ώστε το παιχνίδι να γίνετέ πιο δύσκολο. Γενικότερα η ιδέα του παιχνιδιού είναι πολύ απλή. Απλά σημαδεύουμε το πουλί με τον κέρσορα του ποντικιού που πετάει στην οθόνη και πατώντας αριστερό κλικ πυροβολώντας το. Τώρα κάθε φορά που πετυχαίνουμε το πουλί κερδίζουμε 100 πόντους και κάθε φορά που αστοχούμε παίρνουμε μείον 10 πόντους σαν ποινή και όλα αυτά διαδραματίζονται μέχρι ο χρόνος να φτάσει στα 60 δευτερόλεπτα και τελειώσει το παιχνίδι. Με λίγα λόγια θα δημιουργήσουμε ένα shootingGame σε JavaScript. Χρησιμοποιούμε JavaScript και την createj/slibrary κατά κύριο λόγο και θα προσπαθήσουμε το παιχνίδι που θα φτιάξουμε να μπορεί να τρέχει σε οποιονδήποτε προγράμματα περιήγησης η αλλιώς Browsers.Το παιχνίδι ουσιαστικά θα περιέχει διάφορους ήχους, κινούμενες εικόνες η αλλιώς animations και ένα σύστημα καταγραφής σκορ η αλλιώς scoringsystem.Για να το πετύχουμε αυτό θα χρειαστούμε ένα Cascading Style Sheet αρχείο η αλλιώς ένα CSS όπως και θα το λέμε με το όνομα normalize,ένα HTML αρχείο με το όνομα MainBody, ένα JavaScript η αλλιώς JS αρχείο με το όνομα kodikas και τέλος θα χρειαστούμε ένα φάκελο με το όνομα assets όπου εκεί μέσα θα τοποθετήσουμε όλες τις εικόνες σε μορφή .png και όλους του ήχους που δημιουργήσαμε σε μορφή MP3 και OGG χρησιμοποιώντας το Krita και το Audacity.Για τα τέσσερα αυτά αρχεία λοιπόν θα δούμε πως συνεργάζονται και λειτουργούν μαζί έτσι ώστε να μπορούμε να παίξουμε το παιχνίδι μας.

1.2 Περιγραφή και Οργάνωση

Αρχικά στο κεφάλαιο 2 παρουσιάζουμε την αρχή του παιχνιδιού - Storyboard δηλαδή το πώς ξεκινήσαμε να το δημιουργούμε καθώς και την λειτουργία του κυρίως με εικόνες δηλαδή το πώς συμπεριφέρεται το παιχνίδι.

Στη συνέχεια στο κεφάλαιο 3 περιγράφουμε τα απαραίτητα εργαλεία – προγράμματα που θα χρειαστούμε για να υλοποιήσουμε το παιχνίδι μας τα οποία συμβάλουν σημαντικά σε αυτό, καθώς και την γλώσσα , βιβλιοθήκες που θα χρησιμοποιήσουμε.

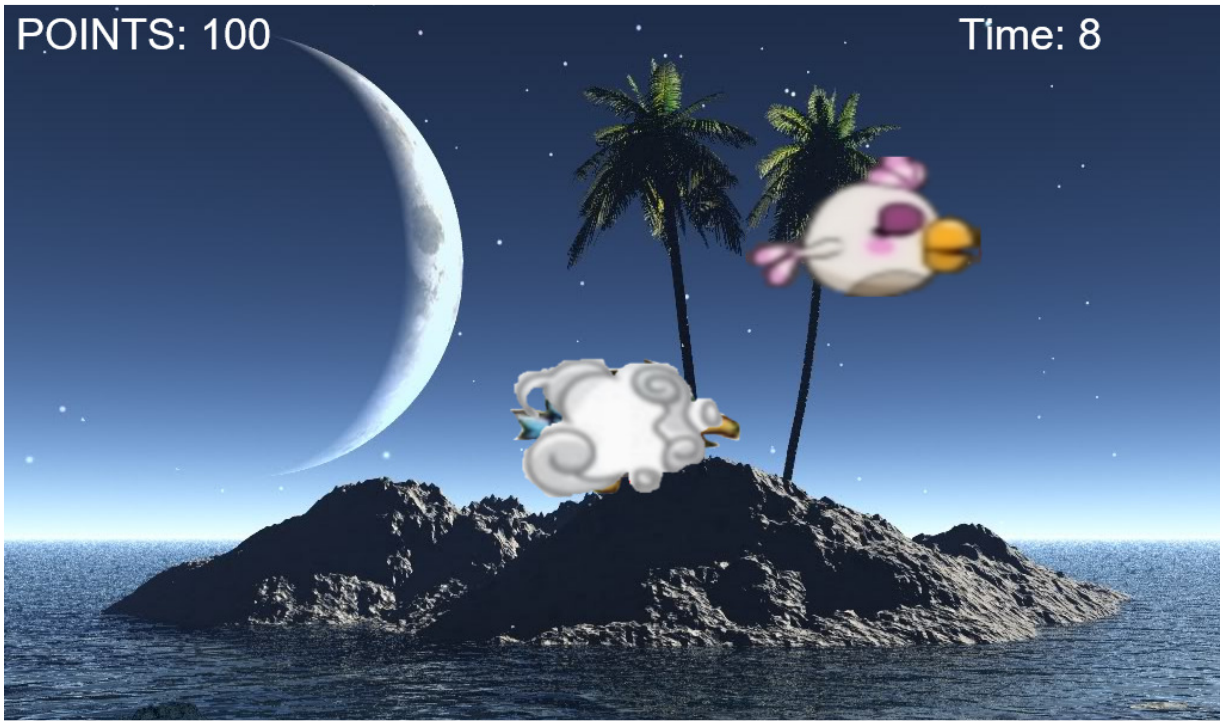
Έπειτα στο κεφάλαιο 4 αναλύουμε τους δυο βασικούς πυλώνες του παιχνιδιού το Normalize.cssκαι toMainBody.html τα οποία συνεργάζονται για την επιτυχημένη λειτουργία του παιχνιδιού.

Τέλος στο κεφάλαιο 5 διεισδύουμε περισσότερο στην καρδιά καθώς και στην επίσημη αρχή του παιχνιδιού δηλαδή στο αρχείο Κώδικας.jsστο οποίο θα δημιουργήσουμε από την αρχή όλο μας το παιχνίδι βήμα – βήμα.

Κεφάλαιο 2– Η αρχή και η λειτουργία του παιχνιδιού

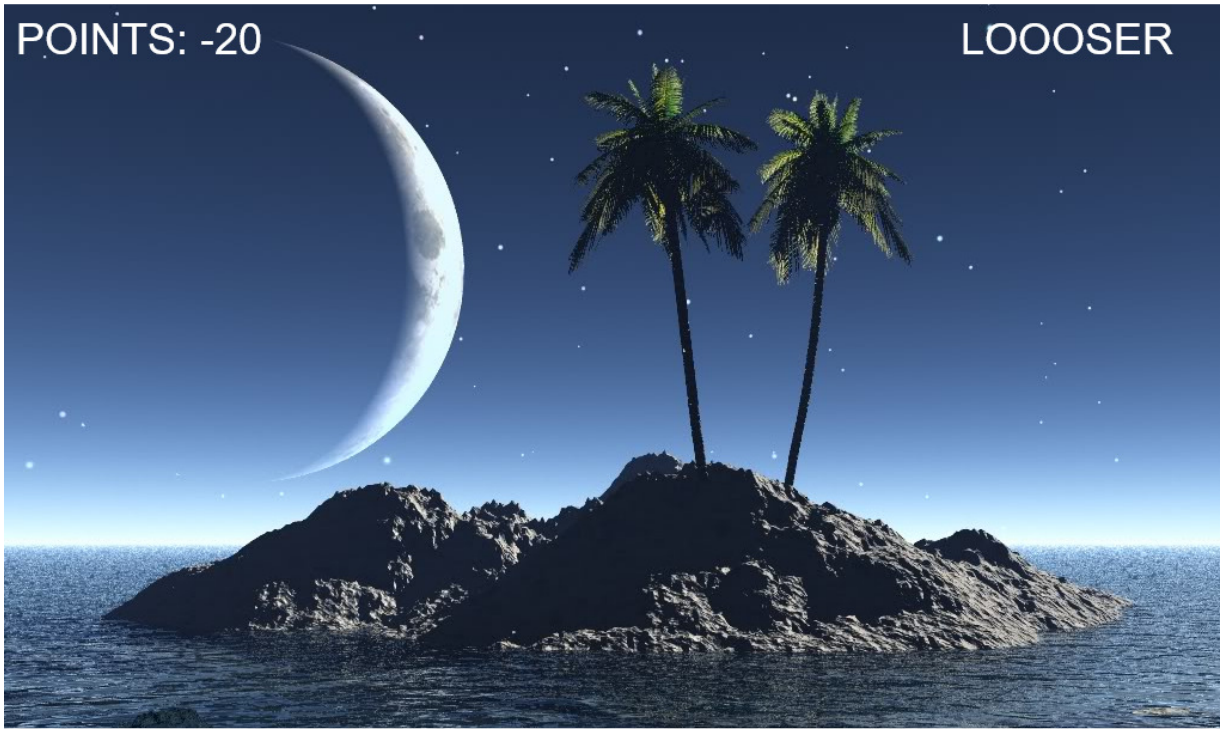
2.1 Η λειτουργία

Η λειτουργία του παιχνιδιού είναι πάρα πολύ απλή. Αρχικά αυτό που κάνει το παιχνίδι είναι να μας "δυσκολεύει" όλο και περισσότερο δηλαδή να πηγαίνει όλο και ποιο γρήγορα τα δύο μας πουλιάμε αποτέλεσμα να μην πτωχαίνουμε πάντα κάποιο από τα πουλιά με συνέπεια να πηγαίνουν όλο και ποιο γρήγορα. Έπειτα μας "βοηθάει " δείχνοντας μας " το που πραγματοποιήθηκε η προηγούμενη μας βολή καθώς και το αν έχουμε πετύχει κάποιο πουλί από το τα δυο (Εικόνα 1 και 2). Τέλος με την λήξη ενός χρονομέτρου το παιχνίδι σταματάει αυτόματα δείχνοντας μας το τελικό αποτέλεσμα (Εικόνα 3).



Εικόνα 1

Εικόνα 2

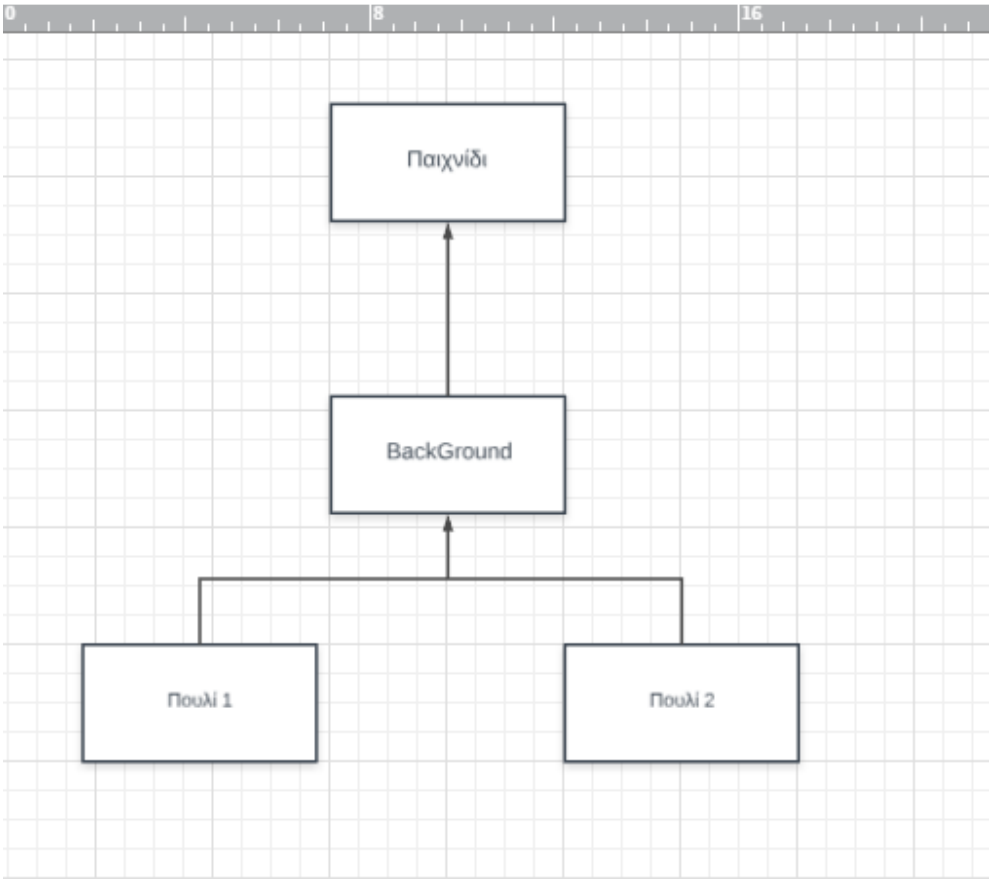


Εικόνα 3

2.2 StoryBoard

Το storyboards είναι ένα ισχυρό εργαλείο για χρήση στο σχεδιασμό παιχνιδιών. Όταν το storyboard το παιχνίδι σας, δημιουργείτε ένα σύνολο εναλλάξιμων καρτών – εικόνων που θα κάνουν μια απλή αναπαράσταση των σκηνών σας με λογική σειρά. Είναι ένας οπτικός τρόπος για να σχεδιάσουμε την ιστορία του παιχνιδιού μας με μια σειρά από "εικόνες" ή άλλα οπτικά στοιχεία, για να ενσωματώσουμε χαρακτήρες, σκηνές, αντικείμενα, τόνο, ενέργειες, λειτουργίες παιχνιδιού και άλλα με συνέπεια να είναι ένα εξαιρετικό εργαλείο στη διαδικασία σχεδιασμού του παιχνιδιού σας. Σκοπός του StoryBoard είναι να δημιουργήσουμε από την αρχή ένα παιχνίδι στον υπολογιστή κατηγορίας (shootingGame) όπου ο παίκτης θα προσπαθεί να πυροβολήσει τα δυο ιπτάμενά πουλιά που θα υπάρχουν και θα πετάνε μέσα στον background μας.

Με στόχο λοιπόν την αρχική μας ιδέα δημιουργήσαμε το πρώτο μας ObjectDiagram με σκοπό την καλύτερη κατανόηση του παιχνιδιού (Βλέπε ObjectDiagram 1).



ObjectDiagram1

Με βάση πάντα λοιπόν την αρχική μας ιδέα ξεκινήσαμε κάνοντας πρώτα τον σχεδιασμό του πρώτου πουλιού μας (πουλί 1) χωρίς να έχουμε καμία επίγνωση το πως θα εξελιχθεί το παιχνίδι στην πορεία. Η αρχική μας ιδέα ήταν απλά ένα πουλί να πετάει χωρίς να ανοιγοκλείνει τα φτερά του. Σκεφτείτε δηλαδή μια εικόνα να μένει σταθερή και απλά να κουνιέται πέρα δώθε στην οθόνη (Figure 1).

Figure1

Έπειτα περάσαμε αυτόματα στον σχεδιασμό του δεύτερου πουλιού (πουλί 2 , Figure 2) όπου και δημιουργήθηκε η ιδέα να κάνουμε και τα δυο πουλιά μας «κανονικά» πουλιά δηλαδή να πετάνε και όχι απλά να είναι δυο εικόνες που απλά θα κουνιούνται μέσα στο BackGround μας. Έτσι καταφέραμε και σχεδιάσαμε το Πουλί 2 σε πέντε διαφορετικές φάσης έτσι ώστε με την βοήθεια του κώδικα ,όπου και θα δημιουργήσουμε ,να τρέχουν αυτές οι φάσης η μια μετά την άλλη με αποτέλεσμα να φαίνεται ότι του πουλί 2 πετάει (Figure 3).

Figure 1

Figure3

Στην συνέχεια έπρεπε και το πουλί 1 με την σειρά του να σχεδιαστεί σε πέντε διαφορετικές φάσης – στάδια για να φαίνεται μέσω το κώδικα, όπως και προ είπαμε ότι πετάει. Έτσι και έγινε! (Figure 4).

Figure4

Τέλος το τελευταίο αντικείμενο - τοπίο που θα χρειαστεί να σχεδιασθούμε είναι το background μας το οποίο προέκυψε από την αγάπη που της έχουμε όλοι, μα φυσικά μια θάλασσά και ένα νησί. Αποφασίσαμε λοιπόν να σχεδιάσουμε ένα νησί στο οποίο θα πετάνε τα πουλιά μας (Figure 5).

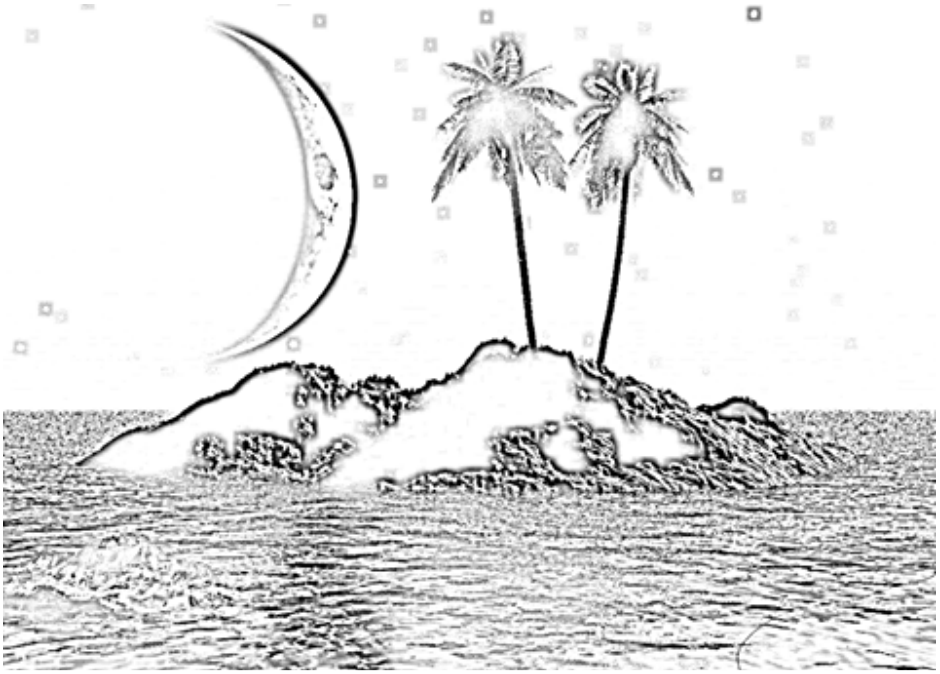
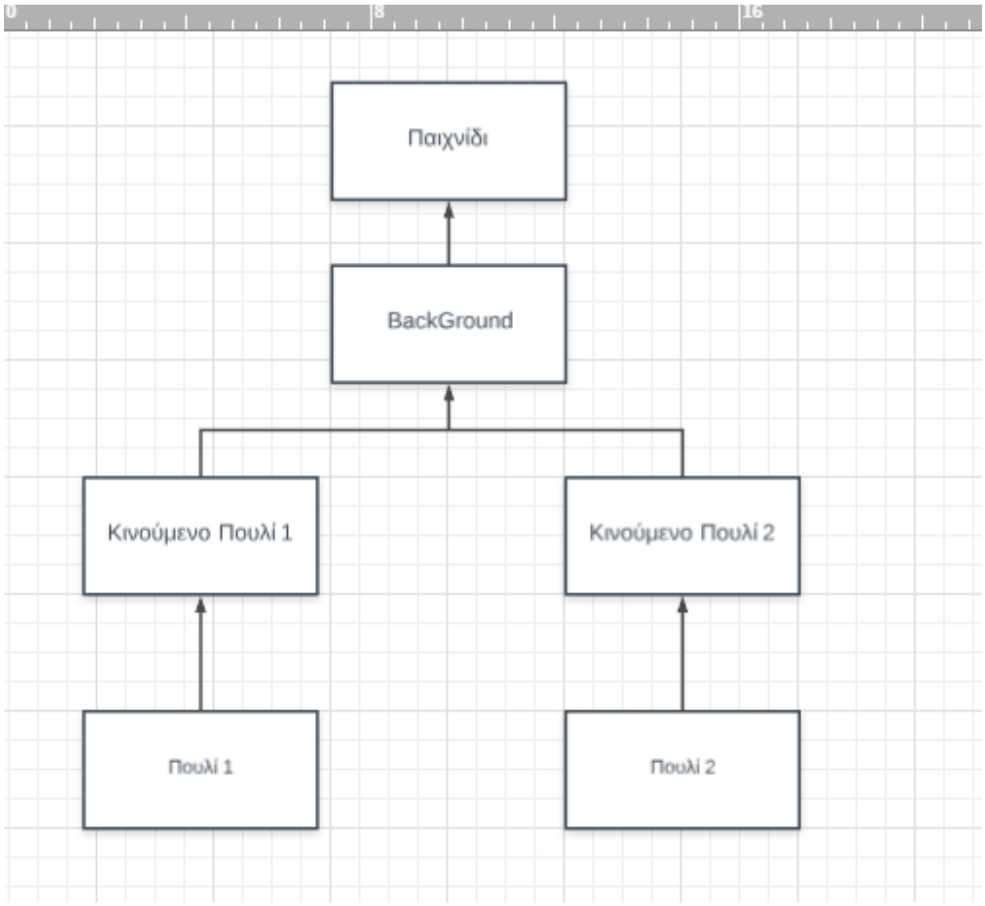


Figure5

Έτσι με την βοήθεια του Krita και με πολύ φαντασία και χρώμα το αποτέλεσμα που προκύπτει φαίνεται στο Figure 6 καθώς και το ObjectDiagram (ObjectDiagram 2).

Figure 2



ObjectDiagram2

Κεφάλαιο 3– Απαραίτητα Εργαλεία

3.1 Krita

Αρχικά για να δημιουργήσουμε τις φωτογραφίες και τα Spritesheets που θα χρειαστούμε για το παιχνίδι μας θα χρειαστούμε ένα πρόγραμμα ζωγραφικής.

Επιλέγουμε το Krita το οποίο είναι ένα πρόγραμμα ζωγραφικής ελεύθερης και ανοικτής πηγής, το οποίο έχει σχεδιαστεί αποκλειστικά για ερασιτέχνες και επαγγελματίες καλλιτέχνες. Με βάση τα περισσότερα σχόλια των χρηστών του, το εργαλείο σχεδίασης και ζωγραφικής περιγράφεται ως μια εναλλακτική λύση για το Photoshop λόγω της πλήρως εξοπλισμένης διεπαφής και της γρήγορης ζωγραφικής λειτουργίας. Ως καλλιτεχνικό εργαλείο, παρέχει μια καινοτόμο λύση για τους καλλιτέχνες με έννοιες και υφή, εικονογράφους και καλλιτέχνες VFX (Visual Effects) στη βιομηχανία κινηματογράφου. Επιπλέον, το πρόγραμμα είναι ιδανικό για 2D ή 3D καλλιτέχνες στη βιομηχανία παιχνιδιών. Χρησιμοποιεί παραδοσιακές τεχνικές ζωγραφικής. αλλά, θεωρείται επίσης ότι έχει την ικανότητα να αντικαταστήσει Corel Painter, καινοτομεί τους προσφέροντας δημιουργικά εργαλεία που λειτουργούν ομαλά με οποιαδήποτε γραφικά δισκία. Το πρόγραμμα ζωγραφικής προσφέρει σύνολα εργαλείων. δηλαδή, εξομάλυνση, καθρέφτισμα, πίεση, μετασχηματισμός, στρωματοποίηση και ανάμιξη χρωμάτων ή ανάμειξη. Το Krita υποστηρίζει επίσης τη χρήση και τον χειρισμό εικόνων HDR ή υψηλής δυναμικής εμβέλειας, οι οποίες είναι ρεαλιστικές φωτογραφίες. Τα οφέλη τώρα που μπορεί να μας παρέχει το Krita είναι τα εξής:

- Είναι χτισμένο με περισσότερους από 9 κινητήρες βούρτσας, το Krita προσφέρει πολλούς τύπους βούρτσας που μπορούν να χρησιμοποιήσουν οι καλλιτέχνες ανάλογα με το ποια είναι τα πιο άνετα. Αυτές οι βούρτσες είναι σχεδιασμένες να εκτελούν μοναδικές και ειδικές λειτουργίες και χρησιμοποιούνται για την κάλυψη συγκεκριμένων αναγκών. Επιπλέον, οι χρήστες θα μπορούν να δημιουργούν τις δικές τους βούρτσες και να τις επισημαίνουν μέσω ενός μοναδικού συστήματος ετικετών.

- Είναι σημαντικό οι χρήστες να είναι, τουλάχιστον, εξοικειωμένοι με τις βασικές βούρτσες στο Krita, ειδικά αν είναι η πρώτη φορά που χρησιμοποιούν το λογισμικό. Ορισμένες από τις βασικές βούρτσες που χρησιμοποιούνται στο Krita είναι το Ink Ballpen, το Fill Brush

Circle και το Fill Brush Block. Υπάρχουν επίσης και είδη βούρτσας που μπορούν να χρησιμοποιηθούν ειδικά για ζωγραφική. όπως το υγρό τρίχας, η υφή των τριχών, το βασικό μείγμα μαλακής βούρτσας και η πίεση του αεροσυμπιεστή.

• Το στυλό μελάνης είναι παρόμοιο με ένα στυλό και αυτό είναι το είδος της βούρτσας που συνιστάται για σκίτσο ή γραφή. Μερικοί καλλιτέχνες προτιμούν να δουλεύουν με σιλουέτες ως σημείο εκκίνησης για τη δημιουργία μιας γραφικής τέχνης. Οι σιλουέτες είναι εικόνες ατόμων ή αντικειμένων που αντιπροσωπεύονται ως στερεά σχήματα ενός μόνο χρώματος, συνήθως μαύρου χρώματος.

• Ο κύκλος βούρτσας γεμίσματος και πλήρωσης βούρτσας του Krita είναι οι τύποι των βούρτσας που έχουν σχεδιαστεί για σιλουέτες και έχουν πάντοτε εκατό τοις εκατό αδιαφάνεια. Αυτές οι βούρτσες έχουν χαρακτηριστικό μέγεθος πίεσης. ως εκ τούτου, αλλάζουν σε μέγεθος ανάλογα με την ποσότητα πίεσης που εφαρμόζεται στο γραφικό ταμπλέτα. Η δημιουργία της μετάβασης μεταξύ των χρωμάτων ή της βαφής / ανάμειξης χρωμάτων μπορεί να γίνει μέσω της βούρτσας Wrist Wet. και, αν οι χρήστες θέλουν να προσθέσουν μια λεία υφή, αυτό είναι δυνατό χρησιμοποιώντας τη βούρτσα υφής Bristles.

• Εν τω μεταξύ, η αφαίρεση των άκρων είναι η λειτουργία της μαλακής βούρτσας Basic Mix. Αυτό οφείλεται στο γεγονός ότι αυτή η βούρτσα έχει την ικανότητα να επιλέγει μόνο μια μικρή ποσότητα χρώματος. Για να προσθέσετε αντίθεση σε ένα σχέδιο, οι χρήστες μπορούν να χρησιμοποιήσουν την πίεση Airbrush. Οι βούρτσες στο Krita προάγουν την ευελιξία και ενισχύουν τη δημιουργικότητα.

• Για να βοηθήσει τους χρήστες να οργανώσουν τα σχέδιά τους, το Krita προσφέρει επίσης ένα εργαλείο που τους επιτρέπει να διαχειρίζονται τα στρώματα. Μέσω αυτής της λειτουργικότητας, θα μπορούν να συνδυάζουν και να ομαδοποιούν σχέδια. Για παράδειγμα, αν θέλουν το σχέδιο σε ένα στρώμα να χρησιμεύσει ως φόντο για ένα άλλο σχέδιο που βρίσκεται σε άλλο επίπεδο, παρέχονται τα εργαλεία διαχείρισης στρώματος του λογισμικού για τη συγχώνευση των δύο σχεδίων. Επιπλέον, εάν θέλουν να αφαιρέσουν μερικά τμήματα ενός σχεδίου, θα μπορούν να το κάνουν χρησιμοποιώντας τις επιλογές προβολής στρώματος.

• Επιπλέον, η Krita προσφέρει υποστήριξη στους χρήστες παρέχοντας εκπαιδευτικό και εκπαιδευτικό υλικό που μπορούν να έχουν πρόσβαση μέσω του Διαδικτύου. Εξοπλίζοντας τους

χρήσιμες πληροφορίες, όπως η χρήση των εργαλείων, οι χρήστες είναι βέβαιοι ότι θα έχουν εκτεταμένη κατάρτιση για να μάθουν τα βασικά και εκ των προτέρων χαρακτηριστικά του προγράμματος. Αυτά τα εκπαιδευτικά υλικά προετοιμάζονται από το ίδιο το Ίδρυμα Krita, επομένως, οι χρήστες έχουν πρόσβαση σε αξιόπιστες και ενημερωμένες πληροφορίες.

3.2 Audacity

Στην συνέχεια το δεύτερο και εξίσου σημαντικό πρόγραμμα που θα χρειαστούμε είναι έναςδωρεάνεπεξεργαστής και συσκευήεγγραφήςήχουπολλαπλών κομματιών για Windows, MacOSX, GNU / Linux και άλλα λειτουργικά συστήματα. Επιλέγουμε το Audacity γιατίπρογράμματα όπως το Audacity ονομάζονται επίσης λογισμικό ανοιχτού κώδικα, επειδή ο πηγαίος κώδικας τους είναι διαθέσιμος για οποιονδήποτε να μελετήσει ή να χρησιμοποιήσει. Υπάρχουν χιλιάδες άλλα προγράμματα ελεύθερου και ανοιχτού κώδικα, όπως το πρόγραμμα περιήγησης ιστού του Firefox, οι σειρές γραφείου LibreOffice ή Apache OpenOffice και ολόκληρα λειτουργικά συστήματα που βασίζονται στο Linux, όπως το Ubuntu. Η διεπαφή μεταφράζεται σε πολλές γλώσσες. Μπορούμε να χρησιμοποιήσουμε το Audacity για να:

- Εγγραφή ζωντανού ήχου.
- Καταγράψτε την αναπαραγωγή του υπολογιστή σε οποιοδήποτε μηχάνημα των Windows Vista ή νεότερης.
- Μετατρέψτε τις κασέτες και τις εγγραφές σε ψηφιακές εγγραφές ή CD.
- Επεξεργαστείτε αρχεία ήχου WAV, AIFF, FLAC, MP2, MP3 ή Ogg Vorbis.

AC3, M4A / M4R (AAC), WMA και άλλες μορφές που υποστηρίζονται χρησιμοποιώντας προαιρετικές βιβλιοθήκες.

- Κόψτε, αντιγράψτε, συνδέστε ή συνδυάστε τους ήχους μαζί.
- Πολυάριθμα εφέ συμπεριλαμβανομένης της αλλαγής της ταχύτητας ή του βήματος μιας εγγραφής.

Εμείς κυρίως θα το χρειαστούμε περισσότερο για να κόψουμε – επεξεργαστούμε αρχεία ήχου . WAV, AIFF, FLAC, MP2, MP3 ή Ogg Vorbis.

AC3, M4A / M4R (AAC), WMA και άλλες μορφές που υποστηρίζονται χρησιμοποιώντας προαιρετικές βιβλιοθήκες.

3.3 Visual Studio Code

Τέλος ένα από τα ποιο σημαντικάίσως και το σημαντικότερο θα έλεγεκανείς είναι το VisualStudioCode. Το Visual Studio Code είναι ένας επεξεργαστής πηγαίου κώδικα που αναπτύχθηκε από τη Microsoft για Windows, Linux και macOS. Περιλαμβάνει υποστήριξη για εντοπισμό σφαλμάτων, ενσωματωμένο έλεγχο Git, επισήμανση σύνταξης, έξυπνη ολοκλήρωση κώδικα, αποσπάσματα και refactoring κώδικα. Είναι επίσης προσαρμόσιμη, ώστε οι χρήστες να μπορούν να αλλάζουν το θέμα του συντάκτη, τις συντομεύσεις του πληκτρολογίου και τις προτιμήσεις. Είναι δωρεάν και ανοικτού κώδικα, αν και η επίσημη λήψη είναι υπό ιδιωτική άδεια. Το Visual StudioCode βασίζεται στο Electron, ένα πλαίσιο που χρησιμοποιείται για την ανάπτυξη εφαρμογών Node.js για την επιφάνεια εργασίας που εκτελείται στη μηχανή εμφάνισης Blink. Παρόλο που χρησιμοποιεί το ηλεκτρονικό πλαίσιο, το λογισμικό δεν χρησιμοποιεί το Atom και χρησιμοποιεί την ίδια συνιστώσα επεξεργαστή (με την κωδική ονομασία "Monaco") που χρησιμοποιείται στα Visual Studio Team Services (πρώην Visual Studio Online).Στην Έρευνα Προγραμματιστών Stack Overflow 2018, ο κώδικας Visual Studio κατατάχθηκε ως το πιο δημοφιλές εργαλείο περιβάλλοντος για προγραμματιστές, με το 34,9% των 75.398 ερωτηθέντων να ισχυρίζονται ότι το χρησιμοποιούν.Ο κώδικας Visual Studio συνδυάζει την απλότητα ενός επεξεργαστή πηγαίου κώδικα με ισχυρά εργαλεία ανάπτυξης, όπως η ολοκλήρωση και η αποσφαλμάτωση κώδικα IntelliSense.

Πρώτα απ’ όλα, είναι ένας συντάκτης που ξεφεύγει από το δρόμο σας. Ο ευχάριστα χωρίς τριβή κύκλος επεξεργασίας-δημιουργίας-εντοπισμού σφαλμάτων σημαίνει λιγότερος χρόνος για το περιβάλλον σας και περισσότερος χρόνος για την εκτέλεση των ιδεών σας. Δευτερον επεξεργασία, δημιουργία και αποδιοργάνωση εντοπισμού σφαλμάτων γινονται με αριστη ευκολία. Τριτον είναι ένα εργαλείο ανάπτυξης κέντρου επεξεργασίας κώδικα που μπορουμε να δημιουργήσουμε εφαρμογές πολλαπλών πλατφόρμων ιστού και cloud.

3.4 Βιβλιοθήκη-Library

Μια βιβλιοθήκη η αλλιώς library (όπως και θα την αναφέρουμε από εδώ και πέρα) είναι μια οργανωμένη συλλογή χρήσιμων λειτουργιών. Μια τυπική βιβλιοθήκη θα μπορούσε να περιλαμβάνει λειτουργίες χειρισμού συμβολοσειρών, ημερομηνιών, στοιχείων HTML DOM, συμβάντων, cookies, animations, αιτημάτων δικτύου (networkrequests) και πολλά άλλα. Κάθε συνάρτηση επιστρέφει τιμές στην εφαρμογή που καλούμε που μπορεί να υλοποιηθεί, ωστόσο, εμείς επιλέγουμε. Μπορούμε να το σκεφτούμε για παράδειγμα και σαν μια επιλογή εξαρτημάτων αυτοκινήτου στην οποία είμαστε ελεύθεροι να χρησιμοποιήσουμε οποιοδήποτε για να βοηθήσουμε στην κατασκευή ενός οχήματος εργασίας αλλά πρέπει να κατασκευάσουμε τον εαυτό μας με τον εαυτό μας.

Οι βιβλιοθήκες παρέχουν κατά κανόνα ένα υψηλότερο επίπεδο αφαίρεσης που εξομαλύνει τις λεπτομέρειες εφαρμογής και ασυνέπειες. Για παράδειγμα, το Ajax βασίζεται κανονικά στο API XMLHttpRequest, αλλά αυτό απαιτεί αρκετές γραμμές κώδικα και υπάρχουν λεπτές διαφορές μεταξύ των Browsers. Μια βιβλιοθήκη μπορεί να παρέχει μια απλούστερη λειτουργία Ajax (), ώστε να είμαστε ελεύθεροι να επικεντρωθούμε στην επιχειρησιακή λογική υψηλού επιπέδου.

3.5 JavaScript Language

Η JavaScript έχει γίνει μία από τις πιο δημοφιλείς γλώσσες προγραμματισμού ηλεκτρονικών υπολογιστών στον Παγκόσμιο Ιστό (Web). Αρχικά, όμως, πολλοί επαγγελματίες προγραμματιστές υποτίμησαν τη γλώσσα διότι το κοινό της ήταν ερασιτέχνες συγγραφείς ιστοσελίδων και όχι επαγγελματίες προγραμματιστές (και μεταξύ άλλων λόγων). Με τη χρήση της τεχνολογίας Ajax, η JavaScript γλώσσα επέστρεψε στο προσκήνιο και έφερε πιο επαγγελματική προσοχή προγραμματισμού. Το αποτέλεσμα ήταν ένα καινοτόμο αντίκτυπο στην εξάπλωση των πλαισίων και των βιβλιοθηκών, τη βελτίωση προγραμματισμού με JavaScript, καθώς και αυξημένη χρήση της JavaScript έξω από τα προγράμματα περιήγησης στο Web.

Κεφάλαιο 4 – Html και Css

4.1 MainBody.html

ΑρχικάπάμεναδούμετονσκελετότουπαιχνιδιούδηλαδήτοαρχείοMainBody.html (Εικόνα 2) μέσα από το οποίο θα τρέχουμε το παιχνίδι σε οποιονδήποτε προγράμματα περιήγησης(η αλλιώς Browsers) θελήσουμε. Είναι σημαντικό αυτό το αρχείο με την χρήση του VisualStudioCode να το μετατρέψουμε, αφού τελειώσουμε, σε HTML γιατί αλλιώς δεν θα μπορέσουμε να τρέξουμε-παίξουμε σωστά το παιχνίδι.

Περνώντας με την σειρά μας μέσα στον κώδικα φυσικά έχουμε DOCTYPE στη γραμμή 1(figure 2) και το head (γραμμή 3 έως 8) στο οποίο έχουμε τέσσερα πράγματα που συμβαίνουν. Πρώτα έχουμε την ετικέτα τίτλου (γραμμή 4) που έχουμε ορίσει εμείς με ποιο όνομα θέλουμε δηλαδή να το βλέπουμε στο BlogTechnology.

Δεύτερον στη γραμμή 5 έχουμε ένα σύνδεσμο (link) που απευθύνεται στο normalize.css (κάτι που θα δούμε λίγο αργότερα) με συνέπεια ότι περιέχει να το τρέχει.

Στη συνέχεια μέσα στο HTML έχουμε ένα linkcode.createjs.com/createjs το οποίο είναι η βιβλιοθήκη που θα χρησιμοποιήσουμε για να δημιουργήσουμε το παιχνίδι(γραμμή 6). Στην ουσία εδώ στη γραμμή 6 του MainBody.html (Εικόνα 2) το μοναδικό πράγμα που έχουμε και ξεχωρίζει είναι η χρήση της βιβλιοθήκης η αλλιώς οι τέσσερις βιβλιοθήκες σε μια. Αρχικά υπάρχει η Easeljsκαι αυτή χρησιμοποιείται για να δουλεύει με το canvas.Υπάρχει η Tweenjs η οποία χρησιμοποιείται κυρίως για Tweening Animations την οποία δεν την χρησιμοποιούμε στο συγκεκριμένο παιχνίδι. Η Soundjs η οποία χρησιμοποιείται και δουλεύει κυρίως με διαφορετικούς τύπους ήχων και η Preloadjs η οποία χρησιμοποιείται για την προ φόρτωση των assets την οποία χρησιμοποιούμε και εμείς στο συγκεκριμένο παιχνίδι με την χρήση του COPYCDNURL (δηλαδή το δίκτυο διανομής περιεχομένου τους) φυσικά, το μειονέκτημα –

πλεονέκτημα του τώρα είναι ότι το application μας πρέπει να συνδεθεί με το διαδίκτυο για να λειτουργήσει.

Έπειτα μέσα στο headστην γραμμή 8 (Εικόνα 2) έχουμε ένα link στο kodikas στο οποίο είναι το αρχείο JavaScript ,όπου και θα γράψουμε, το οποίο μας βοηθάει να τρέξουμε ότι δημιουργήσουμε σε αυτό.Χρησιμοποιώντας το .js με το όνομα kodikas εκτελούμε όλες της εντολές που εμπεριέχονται μέσα σε αυτό δηλαδή τρέχουμε όλα το αρχείο με το όνομα kodikas.Το .js δημιουργήθηκε στην αναζήτηση της ταχύτερης και πιο σύντομης συνάρτησης JavaScript, με έμφαση στις επιδόσεις υπό V8 και Node.js. Παρουσιάζει εξαιρετική απόδοση τόσο για τα Node.js όσο και για τα προγράμματα περιήγησης. Το δεύτερο ήταν το underscore.js το οποίο είχε μια ωραία σχεδιαζόμενη επέκταση φιλική templating λειτουργία. Το .js είναι γρήγορο, μικρό και δεν έχει εξαρτήσεις.

Τέλος έχουμε και την εντολή body (Γραμμή 9 έως 11) στην οποία υπάρχει ο καμβάς και αυτό είναι το είδος της γραφικής επιφάνειας στην οποία το παιχνίδι θα παίζει αφού πούμε ότι το ID εμπεριέχει το myCanvas .

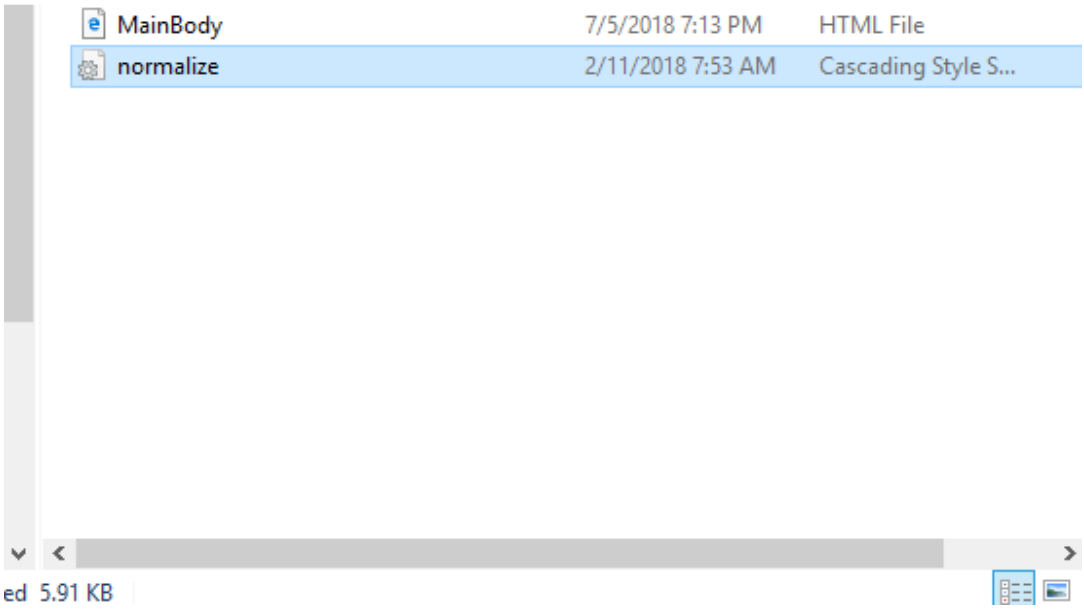
Kodikas.js | MainBody.html

```
1  <!DOCTYPE html>
2  <html>
3      <head>
4          <title>WantToKill....YourTime?</title>
5          <link href="normalize.css" type="text/css" rel="stylesheet" />
6          <script language="javascript" type="text/javascript" src="http://code.createjs.com/createjs-2013.12.12.min.js" ></script>
7          <script language="javascript" type="text/javascript" src="Kodikas.js" ></script>
8      </head>
9      <body>
10         <canvas id="myCanvas"></canvas>
11     </body>
12 </html>
```

Εικόνα 4

4.2 Normalize.css

Τώρα έχουμε το ένα από τα ποιο σημαντικά αρχεία,που πρέπει να χρησιμοποιήσουμε για να τρέξουμε σωστά το παιχνίδι σε οποιονδήποτε browser, το normalizedCSS (Γραμμή 5 από Εικόνα 2 και Εικόνα 3). Με βάση αυτό οι browsers που μπορούν πλέον να υποστηρίξουν το παιχνίδι μας είναι οιChrome, Edge, Firefox ESR +, Internet Explorer 10+, Safari 8+ και Opera. Οι πρωταρχικοί όμως στόχοι του normalize.css είναι να διατηρήσει τις χρήσιμες προεπιλογές προγράμματος περιήγησης αντί να τις διαγράψει, να κανονικοποιήει τα στυλ (Normalizestyles) για ένα ευρύ φάσμα στοιχείων HTML, να διορθώνει τα σφάλματα (CorrectBugs) και τις ασυνέπειες του κοινού προγράμματος περιήγησης και τέλος να βελτιώνει τη χρηστικότητα με τις λεπτές του βελτιώσεις. Στη συνέχεια υποστηρίζει ένα ευρύ φάσμα προγραμμάτων περιήγησης (συμπεριλαμβανομένων των προγραμμάτων περιήγησης για κινητά) και περιλαμβάνει το CSS που εξομοιώνει στοιχεία HTML5, τυπογραφία, λίστες, ενσωματωμένο περιεχόμενο, φόρμες και πίνακες. Παρά το γεγονός όμως ότι το σχέδιο βασίζεται στην αρχή της ομαλοποίησης, χρησιμοποιεί πραγματικές αθετήσεις όπου και προτιμάται. Το Normalize.css διατηρεί χρήσιμες προεπιλογές και διορθώνει τα κοινά σφάλματα. Αρχικά οι επανεκκινήσεις (Resets) επιβάλλουν ένα ομοιόμορφο οπτικό style, χάρη στην συμπίεση των προεπιλεγμένων styles για σχεδόν όλα τα στοιχεία. Αντίθετα, το normalize.css διατηρεί πολλά χρήσιμα προεπιλεγμένα styles προγραμμάτων περιήγησης. Αυτό σημαίνει ότι δεν χρειάζεται να επαναδημιουργήσουμε styles για όλα τα κοινά τυπογραφικά στοιχεία. Όταν ένα στοιχείο έχει διαφορετικά προεπιλεγμένα styles σε διαφορετικά προγράμματα περιήγησης, το normalize.css στοχεύει να καταστήσει αυτά τα styles συνεπή και σύμφωνα με τα σύγχρονα πρότυπα, όταν είναι εφικτό. Τέλος επιδιορθώνει τα κοινά Bugs του προγράμματος περιήγησης για επιτραπέζιους και κινητούς υπολογιστές, τα οποία δεν έχουν δυνατότητα επαναφοράς. Αυτό περιλαμβάνει ρυθμίσεις εμφάνισης για στοιχεία HTML5, διόρθωση μεγέθους γραμματοσειράς για παραμορφωμένο κείμενο, υπερχείλιση SVG στο IE9 και πολλά σφάλματα που σχετίζονται με το περιεχόμενο σε προγράμματα περιήγησης και λειτουργικά συστήματα.


MainBody
7/5/2018 7:13 PM
HTML File
normalize
2/11/2018 7:53 AM
Cascading Style S...
ed 5.91 KB

Εικόνα 5

Κεφάλαιο 5 – Κώδικας.Js

5.1 Variables–Assets

Ολόκληρο το παιχνίδι είναι περίπου (375 σειρές κώδικα) και το έχουμε χωρίσει σε εφτά διαφορετικές λειτουργίες – ενότητες (η αλλιώς κομμάτια του κώδικα) και θα διαπιστώσουμε παρακάτω αναλύοντας τες πώς αυτές συνεργάζονται, δουλεύουν και τρέχουν μαζί. Αρχικά, πριν περάσουμε σε οποιαδήποτε ενότητα πρέπει να έχουμε ορίσει μια σειρά με variables = var (Γραμμή 1 έως 48 ,Εικόνα 4 - Εικόνα 5) που δηλώνονται όλες οι βάσεις ουσιαστικά για να χτίσουμε τις επόμενες ενότητες.

Αρχικά μια πολύ σημαντική εντολή είναι και αυτή στη γραμμή 3 και 4 όπου ρυθμίζω το πλάτος και το ύψος ως σταθερές που είναι το πλάτος και το ύψος στην πραγματική οθόνη του παιχνιδιού και αυτό γιατί σίγουρα ένα από τα πράγματα που ίσως θέλουμε να κάνουμε με τα παιχνίδια μας στο μέλλον είναι να τα δουλέψουμε με μεταβλητά πλάτη και ύψη για αυτό και το αναφέρουμε. Δεδομένου τώρα ότι έχουμε ένα πλάτος και ύψος 1024 από 768 τώρα πρέπει να είναι το ίδιο πλάτος και ύψος με την εικόνα φόντου μας που έχουμε κατασκευάσει μέσω του Krita και του Audacity και βρίσκεται στον φάκελο assets (Εικόνα 6,) ως το συγκεκριμένο asset που είδαμε όταν παίζουμε το παιχνίδι. Επίσης κάποια άλλα πράγματα που θα ασχοληθούμε και θα προγραμματίσουμε όπου και αφορούν τις Εικόνες 4 και 5 είναι η θέση του ποντικιού στον άξονα x και y (γραμμή 5 και 6), τη δημιουργία ενός animation πουλιού και ενός animation αυγού πουλιού με την τεχνική spritesheet (γραμμές 7 έως 15).Έπειτα στις γραμμές 16,17 και στις γραμμές 20,21 υπολογίζουμε και οριστικοποιούμε τη τυχαία θέση του πουλιού και του αυγού πουλιού στον άξονα του χ και y. Στη συνέχεια αρχικοποιούμε τη ταχύτητα του πουλιού καθώς και του αυγού πουλιού στον άξονα του χ και y (γραμμές 18,19 και γραμμές 22,23 αντίστοιχα). Επιπλέον δημιουργούμε το σκορ, το σκορtext καθώς και το χρόνο παιχνιδιού (γραμμές 24 έως 27). Μετέπειτα ορίζουμε το βήμα των δευτερολέπτων που θα αυξάνεται το χρονόμετρο(γραμμή 28),καθορίζουμε το κείμενο του χρονομέτρου (γραμμή 29),σετάρουμε την αρχική αλλά όχι την οριστική θέση του πουλιού καθώς και την θέση του αυγού πουλιού (γραμμές 30 και 31 και γραμμές 32 και 33) και καθορίζουμε το κέντρο του spritesheet για το πουλί και για το αυγό πουλί στον άξονα του χ και y (γραμμές 34 και 35 και γραμμές 36 και 37 αντίστοιχα). Τέλος καθορίζουμε το τέλος του παιχνιδιού στα εξήντα δευτερόλεπτα (γραμμή 38), ορίζουμε τους πόντους που θα λαμβάνουμε σε περίπτωση ευστοχίας ή αστοχίας (γραμμή 39 και

40 αντίστοιχα), τοποθέτηση του score και του χρονομέτρου στους άξονες του χ και y στον καμβά μας (γραμμές 41 και 42 και γραμμές 43 και 44) και δημιουργούμε την αύξηση ταχύτητας των δύο πουλιών επί της εκατό στον άξονα του χ και y (γραμμές 45 και 46) καθώς και την σταδιακή αύξηση δυσκολίας του παιχνιδιού κατά εικοσιπέντε της εκατό στον άξονα του χ και κατά τριάντα της εκατό στον άξονα του y (γραμμές 47 και 48).

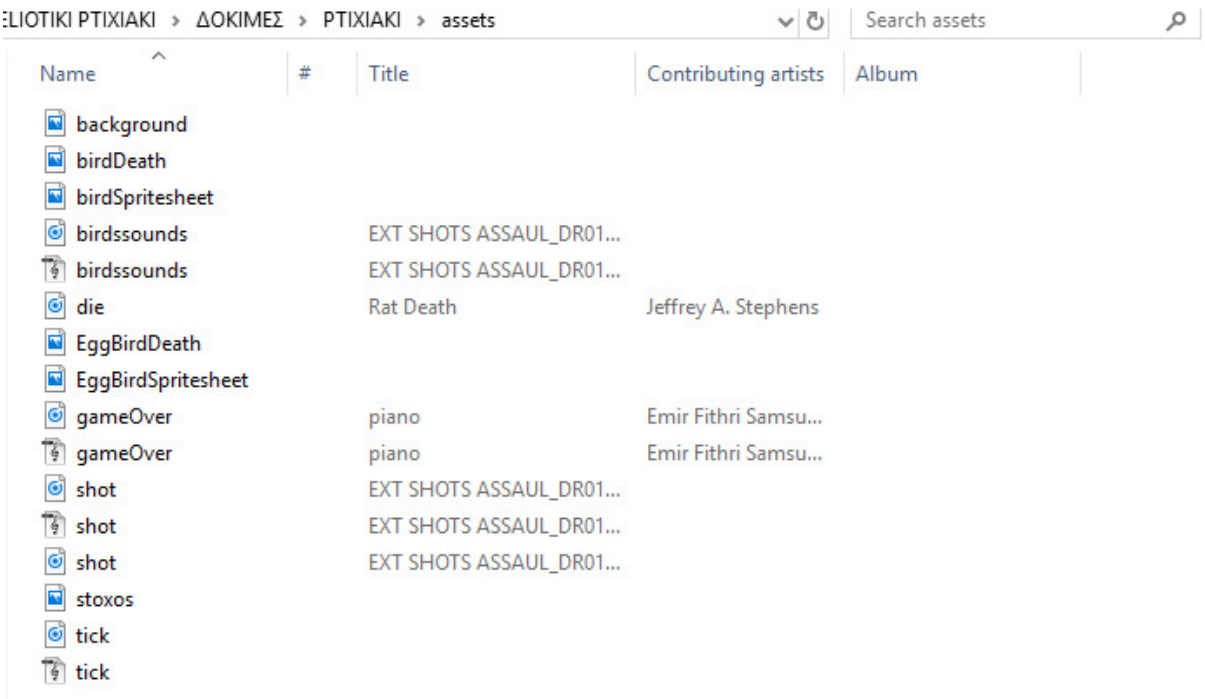
Kodikas.js × MainBody.html

```
1  var context;
2  var queue;
3  var WIDTH = 1024; //Πλατος
4  var HEIGHT = 768; //Υψος
5  var mouseXPosition; //Θεση ποντικιου στο αξονα του X
6  var mouseYPosition; //Θεση ποντικιου στο αξονα του Y
7  var birdImage;  //Εικονα Πουλιου
8  var EggBirdImage; //Εικονα Αυγού-Πουλιου
9  var stage;
10 var animation; //Animation Πουλιου
11 var animation2; //Animation Αυγου-Πουλιου
12 var deathAnimation; //DeathAnimation Πουλιου
13 var deathAnimation2;  //DeathAnimation Αυγου-Πουλιου
14 var spriteSheet;  //spriteSheet πουλιου
15 var spriteSheet2;  //spriteSheet Αυγου-Πουλιου
16 var enemyXPos = Math.floor((Math.random()*1024)+1); //Υπολογισμός οριστικης και τυχαίας θεσης του animation στον αξονα του X
17 var enemyYPos = Math.floor((Math.random()*768)+1);  //Υπολογισμός οριστικης και τυχαίας θεσης του animation στον αξονα του Y
18 var enemyXSpeed = 1.5;   //Ορισμος ταχυτητας του animation στον αξονα του X
19 var enemyYSpeed = 1.75;  //Ορισμος ταχυτητας του animation στον αξονα του Y
20 var enemyXPos2=Math.floor((Math.random()*1024)+1);  //Υπολογισμός οριστικης και τυχαίας θεσης του animation2 στον αξονα του X
21 var enemyYPos2=Math.floor((Math.random()*768)+1);   //Υπολογισμός οριστικης και τυχαίας θεσης του animation2 στον αξονα του Y
22 var enemyXSpeed2 = 1.5;   //Ορισμος ταχυτητας του animation2 στον αξονα του X
23 var enemyYSpeed2 = 1.75;  //Ορισμος ταχυτητας του animation2 στον αξονα του Y
24 var score = 0; //Αρχικοποιηση του score
```

Εικόνα 6

```
25 var scoreText;  //Οριμος κειμενου σκορ
26 var gameTimer;
27 var gameTime = 0;  //Αρχικη τιμη του χρονου
28 var TimePerSecond = 1; //Ορισμος του βηματος των δευτερολέπτων
29 var timerText;  //Οριμος κειμενου χρονομετρου
30 var setPosx = 99; //Σεταρισμα αρχικης θεσης του πουλιου στον αξονα του X
31 var setPosy = 58;  //Σεταρισμα αρχικης θεσης του πουλιου στον αξονα του Y
32 var setPosx2 = 99;  //Σεταρισμα αρχικης θεσης του Αυγου-πουλιου στον αξονα του X
33 var setPosy2 = 58;  //Σεταρισμα αρχικης θεσης του Αυγου-πουλιου στον αξονα του Y
34 var CoSPx = 30;  //Καθοριμος του κεντρου του sprtesheet για το πουλι στον αξονα του X
35 var CoSPy = 60;  //Καθοριμος του κεντρου του sprtesheet για το πουλι στον αξονα του Y
36 var CoSPx2 = 35;  //Καθοριμος του κεντρου του sprtesheet για το Αυγο-πουλι στον αξονα του X
37 var CoSPy2 = 60;   //Καθοριμος του κεντρου του sprtesheet για το Αυγο-πουλι στον αξονα του Y
38 var EndGameInSeconds = 60;  //Καθορισμος του τελους παιχνιδιου στα 60 secs
39 var scoreup = 100; //Ορισμος ποντων σε περιπτωση ευστοχιας
40 var miss = 10;  //Ορισμος ποντων σε περιπτωση αστοχιας
41 var STPossx = 10;  //Καθοριμος θεσης του score στον αξονα του X στον καμβα μας
42 var STPossy = 10;  //Καθοριμος θεσης του score στον αξονα του Y στον καμβα μας
43 var TTxPoss = 800;  //Καθοριμος θεσης χρονομετρου στον αξονα του X στον καμβα μας
44 var TTyPoss = 10;   //Καθοριμος θεσης χρονομετρου στον αξονα του Y στον καμβα μας
45 var PerCentSpeedX = 1.05;  //αυξηση ταχυτητας πουλιου επι τις % στον αξονα του X
46 var PerCentSpeedY =1.06;   //αυξηση ταχυτητας πουλιου επι τις % στον αξονα του Y
47 var PerCentDiffX = 1.25;   //Αυξηση δυσκολιας του παιχνιδιου κατα 25% στον αξονα του X
48 var PerCentDiffY = 1.30;   //Αυξηση δυσκολιας του παιχνιδιου κατα 30% στον αξονα του Y
```

Εικόνα 7



Εικόνα 8

5.2 Window.onload

Η πρώτη λειτουργία που τρέχει είναι η λειτουργία Window.onload και το πρώτο πράγμα που θα κάνουμε εκεί είναι να ρυθμίσουμε τον καμβά μας (τον αναφέρουμε από εδώ και πέρα ως canvasγιατίέτσιακριβώς είναι γραμμένος στον κώδικα)με μέγεθος και ύψος (γραμμές 58 έως 62, Εικόνα 7) ταυτόχρονα πρόκειται να φέρουμε τη βιβλιοθήκη Createj/s καθώς θα κάνουμε αυτό.

Αρχικά στη γραμμή 58 πρέπει να θυμόμαστε ότι ονομάσαμε τον canvas στο HTML " myCanvas "σωστά. Στη γραμμή 10(τουMainBody.htmlαρχείου βλέπε Εικόνα 2) υπάρχει το IDγιαmyCanvas .Έτσι απλά το ανακτάμε εδώ μέσω JavaScript για την ανάθεση με τη μεταβλητή που ονομάζεται canvas.

Όταν δουλεύουμε με τον canvas πρέπει να εργαστούμε σε ένα συγκεκριμένο πλαίσιο(η αλλιώςcontext)(γραμμή 59).

Αφούεπιλέξουμε το πλαίσιο 2d και στη συνέχεια στηγραμμή 60-61 ρυθμίζουμε το πλάτος και το ύψος του καμβά στο πλάτος και στο ύψος που θέσαμε σε αυτές τις σταθερέςπροηγουμένως.

Τώρα στηγραμμή 62 η σκηνή(η αλλιώςstage) είναι μέρος της βιβλιοθήκης createj/s. Τώρα το stage παίρνει τα δεδομένα – πληροφορίες – αντικείμενα (η αλλιώςobjects) τουcanvas μας και δημιουργεί ένα stageobject που είναι παρόμοιο με το πώς το παλιό stage.Τώρα με το stageobject μαςείναι έτοιμοκαι είμαστε σε θέση να επωφεληθούμε από τις μεθόδους και ιδιότητες που είναι διαθέσιμες στο εσωτερικό του createj/s. Έτσι τώρα έχουμε το καμβά μας έτοιμο, διαμορφωμένο και σεταρισμένοσωστά.

```
52  window.onload = function()
53  {
54      /*
55      *      Διαμορφωση του Καμβα σε μεγεθος και υψος
56      *
57      */
58      var canvas = document.getElementById('myCanvas');
59      context = canvas.getContext('2d');
60      context.canvas.width = WIDTH;
61      context.canvas.height = HEIGHT;
62      stage = new createjs.Stage("myCanvas");
```

Εικόνα 9

5.2.1 AssetQueues

Το δεύτεροπράγμα που σετάρουμε και δημιουργούμε- προγραμματίζουμε είναι η ουρά των περιουσιακών στοιχείων(η αλλιώςAssetsQueues)ή η σειρά θα μπορούσαμε να πούμεόπου όλα τα αρχείαήχου -φωτογραφίας πρέπει να φορτωθούν για το παιχνίδι τα οποίαβρίσκονταιστο φάκελοassets(γραμμές 68 έως 71, Εικόνα 8). Τώρα πρέπει να θυμηθούμε ότι μερικές φορές θα παίζουμε το παιχνίδι εξ αποστάσεως και ίσως χρειαστεί λίγος χρόνος για να φορτώσουμε όλα αυτά από το διακομιστή και εμείς θέλουμε να ξεκινήσουμε το παιχνίδι μόλις φορτωθούν όλα.

Έτσι στη γραμμή 68 δημιουργούμε ένα νέο αντικείμενο ουράς και το κάνω με αυτήν τη λέξη-κλειδί (FALSE), επειδή πρόκειται να είναι τοπικό μέρος αυτού του συγκεκριμένου site.

Στη γραμμή 69 εγκαθιστούμε αυτό το plug-in ήχου createj/s στο queue.

Τώρα εδώ στη γραμμή 70 πρόκειται να ασχοληθούμε με το Queuecomplete που σημαίνει ότι ολόκληροτοqueue είναι φορτωμένο(η αλλιώςloaded) και πρόκειται να εκτελέσουμε μια λειτουργία που ονομάζεται queueLoadedοπου και γράψαμε.

Τώρα για το ηχητικό αντικείμενο(η αλλιώς soundobject) πρόκειται επίσης να ορίσουμε μια alternativeExtensions OGG (γραμμή 71), διότι δεν υποστηρίζουν όλα τα προγράμματα περιήγησης(browsers) να παίζουν ένα mp3 αρχείο, που είναι και ο λόγος που έχουμε OGG αρχεία, όπως για παράδειγμα και στον assets φάκελο μας εδώ όπως μπορούμε να δούμε έχουμε το gameOver mp3 και το gameOver OGG(Βλέπε Εικόνα 6).Έτσι με αυτόν τον τρόπο σε προγράμματα περιήγησης που δεν υποστηρίζουν τη μορφή αρχείου mp3 ο ήχος εξακολουθεί να παίζει στο αρχείο OGG.

```
68    queue = new createjs.LoadQueue(false);
69    queue.installPlugin(createjs.Sound);
70    queue.on("complete", queueLoaded, this);
71    createjs.Sound.alternateExtensions = ["ogg"];
```

Εικόνα 10

5.2.2QueueLoadManifest

Το επόμενο κομμάτι του κώδικα που υπάρχει στην πρώτη λειτουργία – ενότητα και είναι πολύ σημαντικό γιατί εδώ στις γραμμές 77-89 (Εικόνα 9)είναι όπου έχουμε ορίσει το δηλωτικό των αντικειμένων που πρόκειται να φορτωθούν καθένα από τα οποία δίνεται ένα αναγνωριστικό(ID)– όνομα που θα χρησιμοποιήσουμε όταν το προσδιορίσουμε αργότερα στο παιχνίδι. Έτσι για δώσουμε και ένα παράδειγμα, την εικόνα φόντου που έχει αντιστοιχιστεί με την πηγή όπου βρίσκεται στην περίπτωσή μας ο φάκελος assets και στη συνέχεια, το όνομα του αρχείου δεξιά.Το κάνουμε αυτό για όλα τα στοιχεία δηλαδή για την εικόνα που απεικονίζεται ο στόχος, για τον ήχο που ακούγεται κάθε φορά που πυροβολάμε, για τον background ήχο που ακούγεται σαν να κελαηδάνε τα πουλιά, για τον ήχο GameOver που ακούγεται όταν τελειώνει το παιχνίδι, για τον ticktack ήχο που ακούγεται καθώς περνάνε τα δευτερόλεπτα στο χρονόμετρο,για τον deathsound ήχο που ακούμε κάθε φορά που πυροβολάμε ένα πουλί και για τα πακέτα sprites μας που φυσικά δείχνουν τα διαφορετικά στάδια των αντικειμένων-πουλιά μας καθώς περνάνε μέσα από τις κινούμενες εικόνες.

```
77    queue.loadManifest([
78        {id: 'backgroundImage', src: 'assets/background.png'},
79        {id: 'stoxos', src: 'assets/stoxos.png'},
80        {id: 'shot', src: 'assets/shot.mp3'},
81        {id: 'background', src: 'assets/birdssounds.mp3'},
82        {id: 'gameOverSound', src: 'assets/gameOver.mp3'},
83        {id: 'tick', src: 'assets/tick.mp3'},
84        {id: 'deathSound', src: 'assets/die.mp3'},
85        {id: 'birdSpritesheet', src: 'assets/birdSpritesheet.png'},
86        {id: 'birdDeath', src: 'assets/birdDeath.png'},
87        {id: 'EggBirdSpritesheet', src: 'assets/EggBirdSpritesheet.png'},
88        {id: 'EggBirdDeath', src: 'assets/EggBirdDeath.png'},
89    ]);
```

Εικόνα 11

5.2.3Queue.Load - GameTimer

Φορτώνουμε την εντολήqueue.load(); (γραμμή 90, Εικόνα 10) έτσι ώστε να φορτώνει όλα αυτά τα στοιχεία στο δηλωτικό ώστε να είναι άμεσα διαθέσιμα. Επίσης δημιουργούμεένα gameTimer στη γραμμή 96 (Εικόνα 11)για ενημέρωσηπρώτον κάθε δευτερόλεπτο επειδή το παιχνίδι είναι 60 δευτερόλεπτα και δεύτερον θέλουμε επίσης να αφήσουμε τον χρήστη να ξέρει πόσο χρονικό διάστημα έχει περάσει στο παιχνίδι, έτσι ώστε να ρυθμίσουμε το gameTimer χρησιμοποιώντας το setInterval.Τώρα το setInterval αυτό που κάνει είναι να τρέχει την updateTime λειτουργία που θα εξετάσουμε και πάλι σε λίγο και τρέχει σε κάθε χίλια milliseconds ή ένα δευτερόλεπτο. Έτσι μας κάνει να θυμηθούμε ότι πριν συζητήσαμε στην γραμμή 70 (ΒλέπεΕικόνα 8) τι γίνεται όταν το queue είναι “complete”.

```
90      queue.load();
```

Εικόνα 12

```
96      gameTimer = setInterval(updateTime, 1000);
```

Εικόνα 13

5.3FunctionQueueLoaded(event)

Έπειτα περνάμε στην queueLoaded function η οποία θα τρέξει αμέσως μετά και θα μας πάει στις γραμμές 100 έως 167 καθώς και στη δεύτερη λειτουργία (Εικόνα 12).Η δεύτερη λειτουργία που τρέχει είναι η λειτουργία- functionqueueLoaded(event).

5.3.1 Προσθήκη Εικόνας Background

Αρχικά θα φροντίσει να πάρει όλα αυτά τα objects της queue και να τα ρυθμίσει στην οθόνη. Τώρα έχουμε την αναφορά του backgroundImage εδώ στη γραμμή 103 η οποία προέρχεται από τη δημιουργία μιας νέας εμφάνισης της δημιουργίας J'sbitmap και το bitmap προέρχεται από την ουρά queue.getResult backgroundImage. Έτσι δουλεύοντας από μέσα προς τα έξω ξεκινάμε εδώ με το Queue.getResult backgroundImage έτσι ώστε να μας πηγαίνει σε εκείνο το backgroundImage ID (βλέπε γραμμή78 Εικόνα 9) φορτωμένο από την queue που είναι assets / background.png τότε δημιουργούμε ένα αντικείμενο createjs.bitmap από αυτό και το ορίζουμε αυτό με var backgroundImage.Τώραστη γραμμή 104το stage.addChild backgroundImageχρησιμοποιείται έτσι ώστε να εμφανίζει φυσικά την εικόνα φόντου που έχουμε δει.

5.3.2Προσθήκηscore - χρονομέτρου

Έπειτα θα φροντίσουμε το κείμενο στην οθόνη που το κείμενο να είναι το σκορ και τοχρονοδιακόπτη στις γραμμές 107 έως 110 και 113 έως 116αντίστοιχα. Έτσι ξεκινάμε με τη γραμμή 107 που εμφανίζει το σκορ. Εδώ ακριβώς δημιουργούμε ένα objectscoreText και έρχεται με createjs.To text που είναι για bitmaptext και το ίδιο το κείμενο περιλαμβάνει το κείμενο ShootToGain συν το ίδιο το σκορ που έχουμε κάνει μια συμβολοσειρά χρησιμοποιώντας JavaScript’s για τη λειτουργία string και στη συνέχεια πώς θέλουμε το κείμενο να εμφανίζεται 36pixelsArial σε White. Στη γραμμή 108 και 109 ρυθμίσαμε τη θέση του κειμένου σύμφωνα με τα αρχικά variables που είχαμε ορίσει στην αρχή με τα STPossx, STPossy όπου 10 επί του άξονα x 10 επί του άξονα y που θα βρίσκεται το κείμενο δηλαδή στην οθόνηκαι έπειτα στη γραμμή 110 προσθέτουμε αυτό το Child στο stage.Έτσιακριβώς το timertext περνάει με την ίδια διαδικασία όπως μπορούμε να δούμε στη γραμμή 113.Γράφουμε τι θέλουμε να φαίνεται στο backgroundμας, παίρνουμε το gameTime και το κάνουμεstringify με το toString και αυτό οφείλεται στο γεγονός ότι αυτή η τιμή είναι αρχικά ένας αριθμός, αλλά το createjs.textαπαιτεί να είναι μια συμβολοσειρά και στη συνέχεια να την ορίσουμε στα ίδια 36 pixelsArial και white. Τώρα στις γραμμές 114 και 115 αυτό θα έχει 800 στον άξονα x και 10 άξονα y (από τα αρχικά variables που είχαμε ορίσει TTxPoss, TTyPoss αντίστοιχα)όπως θα δούμε στην επάνω δεξιά

γωνία της οθόνης όταν παίζουμε το παιχνίδι και τότε στην γραμμή 116 προσθέτουμε το child ξανά στο stage.

5.3.3 Δημιουργία Μουσικής Παιχνιδιού

Μετάπερνάμε στον ήχο του φόντου που ακούσαμε όταν παίζαμε το παιχνίδι είναι αυτό που λίγο ποιο πριν φορτώθηκε στο manifest (βλέπεγραμμή 81, Εικόνα 9).Ονομάζεται background και αποφασίζει να το επαναλάβει ξανά και ξανά και παίζεται με το createjs.sound.play το οποίομπορεί να χρησιμοποιηθεί για οποιοδήποτε ήχο και απλά χρησιμοποιούμε εύκολα τη βιβλιοθήκη createjssoundjs όπως φαίνεται στη γραμμή 119.

```
function queueLoaded(event)
{
    // προσθηκη εικονας background
    var backgroundImage = new createjs.Bitmap(queue.getResult("backgroundImage"))
    stage.addChild(backgroundImage);

    // προσθηκη Score
    scoreText = new createjs.Text("ShootToGain: " + score.toString(), "50px Arial", "#FFF");
    scoreText.x = STPossx;
    scoreText.y = STPossy;
    stage.addChild(scoreText);

    // προσθηκη χρονομετρου
    timerText = new createjs.Text("Time: " + gameTime.toString(), "36px Arial", "#FFF");
    timerText.x = TTxPoss;
    timerText.y = TTyPoss;
    stage.addChild(timerText);

    // Δημιουργια Μουσικης παιχνιδιου
    createjs.Sound.play("background", {loop: -1});


    // Δημιουργια animation του εggπουλιου
    spriteSheet2 = new createjs.SpriteSheet({
        "images": [queue.getResult('EggBirdSpritesheet')],
        "frames": {"width": 198, "height": 117},
        "animations": { "flap2": [0,4] }
    });


     // Δημιουργια animation θανατου του εggπουλιου
    EggBirdDeathSpriteSheet = new createjs.SpriteSheet({
        "images": [queue.getResult('EggBirdDeath')],
        "frames": {"width": 198, "height" : 117},
        "animations": {"die": [0,7, false,1 ] }
    });


    // Δημιουργια animation του πουλιου
    spriteSheet = new createjs.SpriteSheet({
        "images": [queue.getResult('birdSpritesheet')],
        "frames": {"width": 196, "height": 117},
        "animations": { "flap": [0,4] }
    });


    // Δημιουργια animation θανατου του πουλιου
    birdDeathSpriteSheet = new createjs.SpriteSheet({
        "images": [queue.getResult('birdDeath')],
        "frames": {"width": 196, "height" : 117},
        "animations": {"die": [0,7, false,1 ] }
    });


    // Δημιουργια eggbirdspritesheet
    createEnemy2();

    // Δημιουργια birdspritesheet
    createEnemy();

    // Προσθηκη ticker
    createjs.Ticker.setFPS(15);
    createjs.Ticker.addEventListener('tick', stage);
    createjs.Ticker.addEventListener('tick', tickEvent);

    // Σεταρισμα των events αφου φορτωθει το παιχνιδι
    window.onmousedown = handleMouseDown;
}
```

Εικόνα11

5.3.4 Δημιουργία birdspritesheet – eggbirdspritesheet και birddeathspritesheet – eggbirddeathspritesheet

Στη συνέχεια έρχονται δύο πολύ σημαντικά κομμάτια του κώδικα, τα οποίαβρίσκονταιμέσα στην δεύτερηλειτουργία, που πρόκειται να δημιουργήσουμε το birdspritesheet και το eggbirdspritesheet όπου και αφορούντο πέταγμα τουπουλιούκαθώς και του αυγού - πουλιού και το birddeathspritesheetμαζί με το eggbirddeathspritesheetγια όταν πεθαίνει-πυροβολήσουμε το πουλί ή το αυγό – πουλί. Άρα θα επικεντρωθούμε στις γραμμές 139 έως 143 και 147 έως 151 για την δημιουργία του animation πουλιού καθώς και την δημιουργία animation θανάτου του πουλιού (Εικόνα 13και 15αντίστοιχα) και στις γραμμές 123 έως 127 και 131 έως 135 για την δημιουργία του αυγού – πουλιού καθώς και την δημιουργία animation θανάτου του αυγού – πουλιού (Εικόνα 14 και 16 αντίστοιχα).

Αρχικά το birdspritesheetκαθώς και το eggbirdspritesheet έχει 5 στάδια και στην ουσία μας δείχνει ταφτεράσε διαφορετικέςφάσεις και όταν τα βλέπει κανείς να τρέχουνμέσα στον κώδικα μαζί μοιάζει με πουλιά πουπετάνε. Πρώτον θέλουμε να πούμε στο πρόγραμμά μας πόσα διαφορετικά στοιχεία – animations- εικόνες έχουμε στα κινούμενασχέδια, και με τι σειρά να τα κάνουμεanimateγια αυτό έπρεπε να φέρουμε το spritesheet στο πρόγραμμα και δεύτερονκαλό θα ήταν να μπορούμε να βλέπουμετιςεικόνεςπου έχουμεφτιάξει μέσω του Krita (βλέπε Εικόνα 17 και 18)καθώς το κάνουμε αυτό.

Πρώτα από όλα στην γραμμή 139 από Εικόνα 13 το ονομάζουμε spritesheet και χρησιμοποιούμε το createjs.Spritesheet με αποτέλεσμα ο κατασκευαστής της δημιουργίας ακριβώς ενός spritesheet παίρνει όλες τις ρυθμίσεις για το ίδιο το spritesheet . Έτσι με αυτόν τον τρόπο, οι εικόνες για το spritesheet που φορτώσαμε με το preloader νωρίτερα,χρησιμοποιώντας πάλι το queue.getResults,φορτώνοντας tobirdspritesheet στο τέλος της γραμμής140 και αυτό φορτώνει την εικόνα μας που έχουμε στο ενεργητικό του φακέλου. Ύστερα ρυθμίζουμε το πλάτος και το ύψος του κάθε πλαισίου - φωτογραφίαστη γραμμή 141.Κάθε πλαίσιο-φωτογραφίαείναι 196pixels πλάτος και 117 pixelsύψος. Εδώπρέπει να είμαστε προσεκτικοίδιότιπρέπει να ορίσουμετον ίδιο σωστό αριθμό με εκείνο του πλαισίουπου είχαμεορίσεικαθώςφτιάχναμε τα spritesheetsμέσω του Krita(δηλαδή τις αποστάσεις από το ένα spritesheet στο άλλο)γιατί τα animationμας μπορεί να μην εμφανίζονται η να εμφανίζονται λάθος.Και τέλος τα animationsστη γραμμή 142 που πρόκειται να καλέσουμε τα ονομάζουμεflap

και το ορίζουμενα πάει από το πλαίσιο 0 έως το πλαίσιο 4 για αυτό είναι ευρετηριασμένο(δηλαδή ο αριθμός 0 αντιστοιχεί στο πρώτοspritesheet,ο 1 αντιστοιχεί στο δεύτερο και ούτω καθεξής).Με αυτόν τον τρόποπου διαχωρίσαμε τα animations μαςθα μπορούσαμε να έχουμε πολλαπλές κινούμενες εικόνες, ώστε να έχουμε κι άλλο animation στο πλαίσιο 5 έως 10 και να είναι αυτό το birdspritesheet.Άρασυμπεραίνουμε με λίγαλόγια ότι από την γραμμή 139έως142ασχολούμαστε με την δημιουργίαanimationτου πουλιού.

Με τον ίδιο ακριβώς τρόποόπως μπορούμε να δούμε και από την Εικόνα 14 δημιουργούμε και το δεύτεροanimation το EggBirdSpriteSheet (γραμμές 123 έως και 127) με την μόνη διαφορά στην γραμμή 124 και γραμμή 125.

Αρχικά στην γραμμή 124 το μοναδικόπράγμα που πρέπει να αλλάξουμε είναι ότι οι εικόνες για το spritesheet2 που φορτώσαμε με το preloader νωρίτερα,χρησιμοποιώντας πάλι το queue.getResults ,φορτώνοντας το Eggbirdspritesheet στο τέλος της γραμμής. Τέλος στην γραμμή 125 πρέπει να ρυθμίσουμε το πλάτος και το ύψος του κάθε πλαισίου - φωτογραφία διότι κάθε πλαίσιο – φωτογραφία στο συγκεκριμένο spritesheetείναι 198pixels πλάτος και 117 pixels ύψος και πρέπει να είμαστε προσεκτικοί για να ορίσουμε τον ίδιο σωστό αριθμό με εκείνο του πλαισίου που είχαμε ορίσει καθώς φτιάχναμε τα spritesheets μέσω του Krita.

```
138    // Δημιουργια animation του πουλιου
139    spriteSheet = new createjs.SpriteSheet({
140        "images": [queue.getResult('birdSpritesheet')],
141        "frames": {"width": 196, "height": 117},
142        "animations": { "flap": [0,4] }
143    });
```

Εικόνα 14

```
122    // Δημιουργια animation του eggπουλιου
123    spriteSheet2 = new createjs.SpriteSheet({
124        "images": [queue.getResult('EggBirdSpritesheet')],
125        "frames": {"width": 198, "height": 117},
126        "animations": { "flap2": [0,4] }
127    });
```

Εικόνα 15

Στην συνέχεια περνάμε στην δημιουργία του τρίτουanimation που έχουμε στις γραμμές 147έως 151 όπως φαίνεται και στην Εικόνα15το οποίο είναι το birddeathspritesheet και αυτό είναι όταν το πουλίπυροβολείτε και τα διαφορετικά στάδια που έχουμε.

Εδώλοιπόν στις γραμμές 148 και 147 με τον ίδιοακριβώςτρόπο όπως κάναμε και για το Spritesheetκαι για το Spritesheet2 προηγουμένως στιςγραμμές 139 και 123 αντίστοιχα(ΒλέπεΕικόνα 13 και 14) έτσι και εδώ το birdDeath.Τα πλαίσια τα οποίαορίζουμεστην γραμμή 149 είναι 196pixelsπλάτος με 117pixelsύψος σύμφωνα πάντα με τις διαστάσεις που έχουν καθοριστεί από το Krita.

Έπειτα στην γραμμή 150 έχουμε το animations, το ονομάζουμε die και είναι από το 0 έως το πλαίσιο 7.Δεν θέλουμε να επαναλάβει- λουπάρει αυτή τη φορά έτσι έχουμε False και δίπλα του τον αριθμό 1.Αυτόγίνεται έτσι ώστε να ορίζει τα spritesheets που πρόκειται να χρησιμοποιήσουμε με αποτέλεσμα όταν εμφανίζονται θα εμφανιστούν κινούμενα.

Τέλος με ακριβώςτώρα την ίδιαλογική και τρόπο όπως κάναμε για το πρώτο, το δεύτερο και τοτρίτοanimationδημιουργούμε και το τέταρτο και τελευταίοanimation όπου και έχουμε στις γραμμές 131 έως 135 το EggBirdDeathSpriteSheet όπως φαίνεται και στην Εικόνα 16.

Ξεκινώντας λοιπόν από τις γραμμές 131 και 132 το μοναδικό πράγμα που πρέπει να φτιάξουμε είναι οι εικόνες για το EggBirdDeathSpriteSheet που φορτώσαμε με το preloader νωρίτερα, χρησιμοποιώντας πάλι το queue.getResults ,φορτώνοντας το Eggbirddeath στο τέλος της γραμμής.

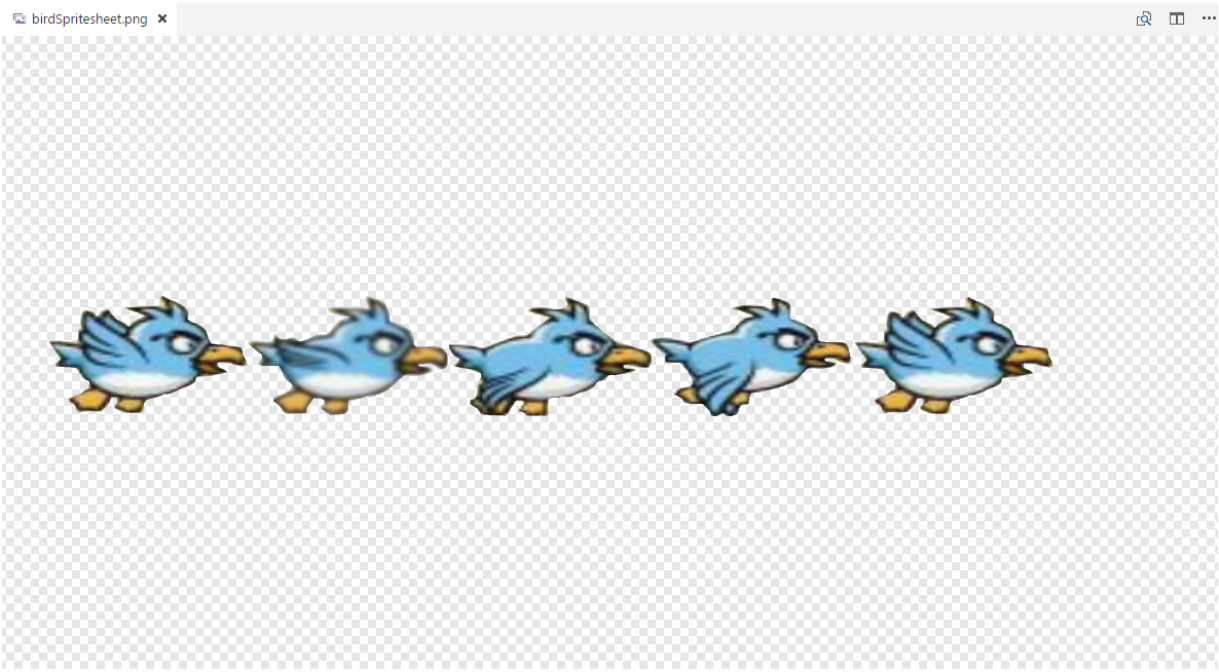
Τα πλαίσια τώρα τα οποία ορίζουμε στην γραμμή 133 είναι 198 pixels πλάτος με 117 pixels ύψος σύμφωνα πάντα με τις διαστάσεις που έχουμε καθορίσει μέσα από το Krita.Τέλος στην γραμμή 134 έχουμε το animations το οποίο ξαναονομάζουμε die και είναι από το 0 έως το πλαίσιο 7. Δεν θέλουμε να ξαναεπαναλάβει - λουπάρει έτσι αυτή τη φορά έχουμε False και δίπλα του τον αριθμό 1. Αυτό γίνεται έτσι ώστε να ορίζει τα spritesheets που πρόκειται να χρησιμοποιήσουμε με αποτέλεσμα όταν εμφανίζονται θα εμφανιστούν κινούμενα.

```
146     // Δημιουργια animation θανατου του πουλιου
147     birdDeathSpriteSheet = new createjs.SpriteSheet({
148         "images": [queue.getResult('birdDeath')],
149         "frames": {"width": 196, "height" : 117},
150         "animations": {"die": [0,7, false,1 ] }
151     });
```

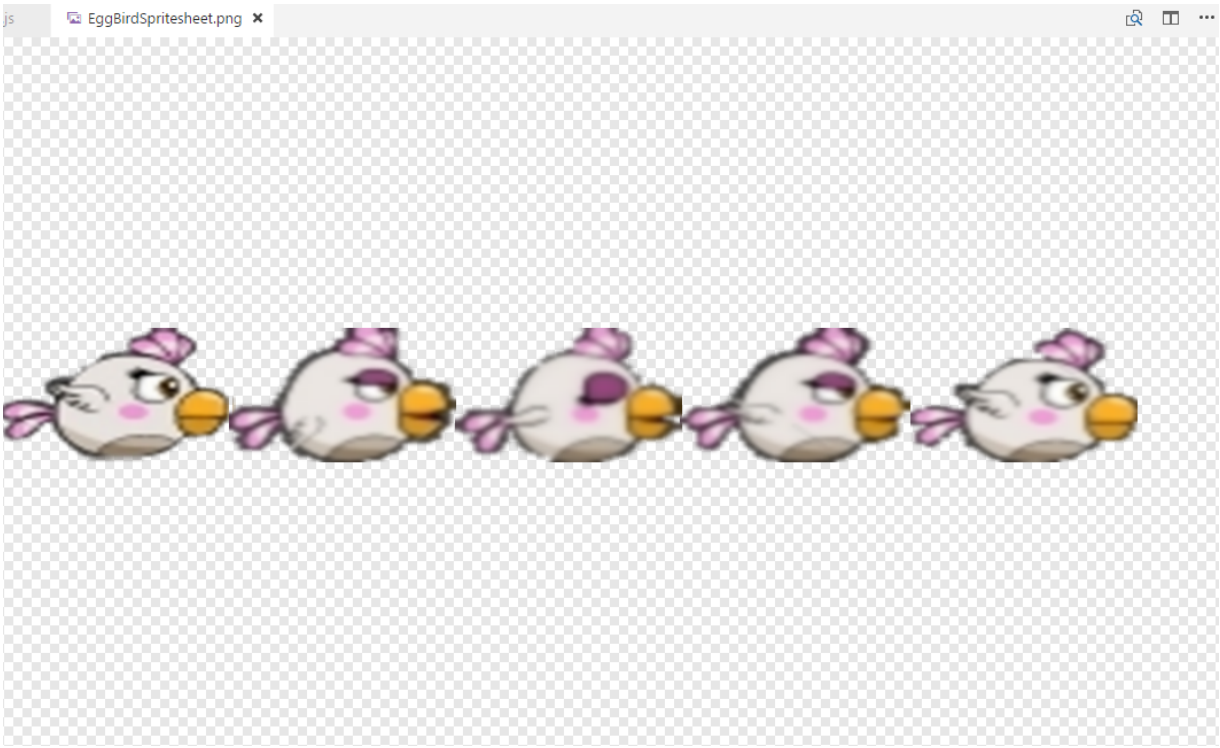
Εικόνα 16

```
130     // Δημιουργια animation θανατου του εγγπουλιου
131     EggBirdDeathSpriteSheet = new createjs.SpriteSheet({
132         "images": [queue.getResult('EggBirdDeath')],
133         "frames": {"width": 198, "height" : 117},
134         "animations": {"die": [0,7, false,1 ] }
135     });
```

Εικόνα 17



Εικόνα 18



Εικόνα 19

5.3.5 CreateEnemy – CreateEnemy2

Τώρα στις γραμμές 158 και 155(στιςοποίες και θα εμβαθύνουμελίγοαργότερα)για να δημιουργήσουμε το birdspritesheet και το eggbirdspritesheet και να τα κάνουμε να εμφανιστούν στην οθόνη το κάνουμεχρησιμοποιώντας την εντολήπου ονομάζεταιcreateEnemy και createEnemy2 (Εικόνα 19 και Εικόνα 20 αντίστοιχα)περνώνταςαυτόματα στο τέλος της δεύτερηςλειτουργίας. Τώρα για τις γραμμές 161έως 163 και γραμμή 166 που δεν αναφέραμε θα τις δούμε στηνπέμπτη και στην έκτηενότητααντίστοιχα.

```
157 | // Δημιουργια birdspritesheet
158 | createEnemy();
```

Εικόνα 20

```
154 | // Δημιουργια eggbirdspritesheet
155 | createEnemy2();
```

Εικόνα19

5.4Δημιουργία birdspritesheet – animation και eggbirdspritesheet – animation2

Εδώ λοιπόν στις γραμμές 170 έως 192 (ΒλέπεΕικόνα21) περνάμε αυτόματα στο τρίτο κομμάτι του κώδικαστο createEnemy και στο createEnemy2.Αρχικά στο createEnemy (γραμμές 182 έως 192) θέτουμε μια μεταβλητή κινούμενη δηλαδήτο animation το οποίοισοδυναμεί με το createjs.sprite από το spritesheet και χρησιμοποιεί τις κινούμενες εικόνες που εμείς μόλις ορίσαμε σε 'flap'(από τον ήχο που κάνουν τα φτερά στην πραγματικότητα όχι στο παιχνίδι).Έτσι η κινούμενη εικόνα είναι τώρα το sprite.Με βάση αυτό το spritesheet που είναι μέσα με το κλουβί animationέχουμε θέσει τα σημεία αποκατάστασης στη μέση του αντικειμένου toregistrationX και το registrationY (τα οποία τα είχαμε ορίσει στην αρχή του κώδικα μας στα variablesως setPosx, setPosy)όπως μπορούμε να δούμε στις γραμμές 186-187.Θέσαμε επίσηςτη θέση στις γραμμές 188 και 189 με βάση το enemyXposition και το enemyYpositionοι οποίεςαν και είναι οι θέσεις στον άξονα x και y υπολογίζονται τυχαία από τα αρχικά variables που έχουμε ορίσει με την συνάρτηση Math.floorkaiMath.random.

Τώρα στη γραμμή 190χρησιμοποιούμε την εντολήgottoandplay η οποία απαιτεί ένα όνομα πλαισίου ή έναν αριθμό στην δική μας περίπτωσηόνομαπλαισίου (flap), έτσι να γνωρίζει πού να στείλει το playhead. Όταν τοποθετούνται σε ένα πλαίσιο, οι ενέργειες αυτές στέλνουν την κεφαλίδα αναπαραγωγής προς τα εμπρός ή προς τα πίσω στο καθορισμένο πλαίσιο και σταματούν ή συνεχίζουν την αναπαραγωγή από εκείνο το σημείο.

Τέλος προσθέτουμε στο stage, χρησιμοποιώντας addChildAt, το animation και προσθέτουμε σε αυτό τον αριθμό 1.Ο λόγος που το κάνουμε αυτό είναι για να το κρατήσουμε πίσω διότι ο χρήστης κινείται γύρωγύρω έτσι χρησιμοποιούμε addChildAtto οποίο είναι ουσιαστικά ο σημαντικότεροςπαράγοντας έτσι ώστε να δημιουργεί τον εχθρό.

Έτσι ακριβώς με τον ίδιο τρόπο δημιουργούμε και το createEnemy2 (γραμμές 170 έως 179) και θέτουμε μια μεταβλητή κινούμενη δηλαδή το animation2 το οποίο ισοδυναμεί με το createjs.sprite από το spritesheet2 και χρησιμοποιεί τις κινούμενες εικόνες που εμείς ορίσαμε σε 'flap2'.Έτσι περνώντας και στις γραμμές 173 και 174 τα σημεία αποκατάστασης στη μέση του αυγού – πουλιού είναι αυτά που είχαμε ορίσει ξανά στην αρχή του κώδικα μας στα variables ως setPosx2, setPosy2 (99 στον άξονα του χ και 58 στον άξονα του y).

Ο υπολογισμός της θέσης του αυγού – πουλιού γίνεταιξανάτυχαίαγιατίέτσι το είχαμεορίσειεξαρχής το enemyXPos2, enemyXPos2 στην αρχή του παιχνιδιού από τα variables (γραμμές 175 και 176).Τώρα στη γραμμή 177 χρησιμοποιούμε ξανά την εντολή gottoandplay η οποία απαιτεί ένα όνομα πλαισίου στην δική μας περίπτωση όνομα πλαισίου (flap2), έτσι να γνωρίζει πού να στείλει το playhead.

Τέλος ξανά με την χρήση του αριθμού 1 και αφού προσθέσουμε στο stage, χρησιμοποιώντας addChildAt, το animation2 (γραμμή 178) δημιουργούμε τον εχθρό η αλλιώς το createEnemy2 η αλλιώς το eggbirdspritesheet – animation2.

```
169  // Δημιουργια eggbirdspritesheet-animation 2
170  function createEnemy2()
171  {
172      animation2 = new createjs.Sprite(spriteSheet2, "flap2");
173      animation2.regX = setPosx2;
174      animation2.regY = setPosy2;
175      animation2.x = enemyXPos2;
176      animation2.y = enemyYPos2;
177      animation2.gotoAndPlay("flap2");
178      stage.addChildAt(animation2,1);
179  }
180
181  // Δημιουργια birdspritesheet - animation
182  function createEnemy()
183  {
184
185      animation = new createjs.Sprite(spriteSheet, "flap");
186      animation.regX = setPosx;
187      animation.regY = setPosy;
188      animation.x = enemyXPos;
189      animation.y = enemyYPos;
190      animation.gotoAndPlay("flap");
191      stage.addChildAt(animation,1);
192  }
```

Εικόνα20

5.5Δημιουργία deathspritesheet – deathanimation και deathspritesheet2 – deathanimation2

Στην συνέχειαπροχωράμε στο τέταρτοκομμάτι του κώδικαπου χωρίζεται στο BirdDeathanimation και στο Eggbirddeathanimation το οποίοέχειάμεσηεπαφή – σχέση με την έκτηενότητα στην οποία θα αναφερθούμε και παρακάτω εκ των υστέρων (γραμμές207έως216 και γραμμές 195 έως 204 αντίστοιχα, Εικόνα22).Το πως θα δημιουργήσουμε τώρα αυτά τα διαφορετικάDeathanimation θα το κάνουμεσύμφωναπάντα με το τρίτο κομμάτι του κώδικα όπως και αναφερθήκαμε προηγουμένως.

Αρχικά θα δούμε την birdDeatheντολή(γραμμή 207) η οποία κατά κύριολόγοτρέχει - χρησιμοποιεί το spritesheet με το ίδιο τρόπο όπως και είδαμε στην τρίτη ενότητα.

Έτσι δημιουργούμε ένα deathAnimation το οποίο είναι sprite με βάση το birdDeathSpriteSheet και το animationμέσα σε αυτό λέγετedie.Θέτουμε το σημείο καταχώρησης το RegistrationPoint (όπου και έχουμε ορίσει στην αρχή τα variables σαν setPosx, setPosy να ισούνται με 99 και 58 αντίστοιχα) στη μέση(βλέπεγραμμή 210 και 211),δηλαδή θέσαμε τη θέση αυτής της κινούμενης εικόνας στην τελευταία γνωστή θέση που ήταν πριν γιατί θέλουμε αυτό να τρέξει στο ίδιο μέρος όπου το πουλί πυροβολήθηκε.

Στη γραμμή τώρα 212 και 213 έχουμε μια birdDeathAnimation που στην ουσία δημιουργεί το κινούμενο σχέδιο τυχαία ,ξανά με βάση τα αρχικάvariables που έχουμεδηλώσει το enemyXPos και το enemyYPos, για όταν το πουλί μας είναι νεκρόδηλαδή με άλλα λόγια το πυροβολήσουμε όπως και γίνεται όταν παίζουμε το παιχνίδι. Έπειτα στο Deathanimationχρησιμοποιούμε και προσθέτουμε το gotoAndPlay και έπειταπροσθέτουμε τη λέξηdie(γραμμή 214).

Τέλος αφού το κάνουμε αυτό παίρνουμε το Deathanimation και το προσθέτουμε αυτό στο stage χρησιμοποιώντας addChild έτσι ώστε να τρέχει το animation θανάτου του πουλιού(γραμμή 215).

Έτσι συμπεραίνοντας αυτό που μπορούμε να κατανοήσουμε είναι ότι το Deathanimation2 η αλλιώς το Eggbirddeathanimation(γραμμές 195 έως 204) δημιουργείτεακριβώς με την ιδιάφιλοσοφία με αποτέλεσμα στις γραμμές 198 και 199 τα

RegistrationPoint του X και Y να ισούνται με τα setPosx2, setPosy2 αντίστοιχαοπού και τα έχουμεπρώτο δηλώσει στα αρχικάvariables.

Στην συνέχεια στις γραμμές 200 και 201 το αυγό πουλί δημιουργείτε τυχαία, ξανά από τα αρχικάvariables που είχαμεδηλώσει, για όταν είναι νεκρό – πυροβολείτε. Στην γραμμή 202 τώρα χρησιμοποιούμε ξανά την εντολή gottoandplay η οποία απαιτεί ένα όνομα πλαισίου στην δική μας περίπτωση όνομα πλαισίου τη λέξη die (ίδιο με την γραμμή 214) για να γνωρίζει πού να στείλει το playhead.

Τέλος αφού το κάνουμε αυτό παίρνουμε το Deathanimation2 και το προσθέτουμε αυτό στο stage χρησιμοποιώντας addChild έτσι ώστε να τρέχει το animation θανάτου του αυγού - πουλιού(γραμμή 203).

```
194  // Δημιουργια deathspritesheet2 -deathanimation 2
195  function EggBirdDeath()
196  {
197    deathAnimation2 = new createjs.Sprite(EggBirdDeathSpriteSheet, "die");
198    deathAnimation2.regX = setPosx2;
199    deathAnimation2.regY = setPosy2;
200    deathAnimation2.x = enemyXPos2;
201    deathAnimation2.y = enemyYPos2;
202    deathAnimation2.gotoAndPlay("die");
203    stage.addChild(deathAnimation2);
204  }
205
206  // Δημιουργια deathspritesheet -deathanimation 2
207  function birdDeath()
208  {
209    deathAnimation = new createjs.Sprite(birdDeathSpriteSheet, "die");
210    deathAnimation.regX = setPosx;
211    deathAnimation.regY = setPosy;
212    deathAnimation.x = enemyXPos;
213    deathAnimation.y = enemyYPos;
214    deathAnimation.gotoAndPlay("die");
215    stage.addChild(deathAnimation);
216  }
```

Εικόνα 21

5.6TickEvent

Έπειταπροχωρώντας στο πέμπτο κομμάτι του κώδικα, θα κοιτάξουμε την καρδιά του παιχνιδιού και θα απαντήσουμε στο χρήστη καθώς πυροβολάμε τοπουλίη το αυγό – πουλί πως κουνιέται σε όλη την οθόνη το οποίο πρόκειται να εξετάσουμε με μια πολύ εύκολη τεχνική

ανίχνευσης - σύγκρουσης χρησιμοποιώντας totickevent που πρωτοεμφανίσαμε στις γραμμές161έως 163 (ΒλέπεΕικόνα 23) και θα χρησιμοποιήσουμε το tickevent, που είχαμεπροσθέσεισε αυτές,στις γραμμές219έως το 264 (Εικόνα 24) για να κάνουμε τοπουλίκαθώς και το αυγό – πουλί να κουνιούνταικαι να βρίσκονται συνέχεια εντός των ορίων του παιχνιδιού – καμβά μας.

5.6.1 Προσθήκη Ticker

Περνώνταςαρχικά στη Εικόνα 23 στο tickerevent τώρα στις γραμμές 161 έως 163 και συγκεκριμένα στην 161 μας λέει ουσιαστικά πόσο γρήγορα θα animate.Έτσι θέτουμε 15 καρέ ανά δευτερόλεπτο (Framespersecond)και στη συνέχεια προσθέτουμε ένα tick στο stage στην γραμμή 162 και στη συνέχεια προσθέτουμε στο tick κάτι που θα ονομάσουμε το tickevent στη γραμμή 163. Έτσι, κάθε φορά που κάνει tick 15 καρέ ανά δευτερόλεπτο το stage θα κάνει update(θα ενημερώνεται δηλαδή) και θα καλεί αυτόματα το tickevent. Έτσι κατά τη διάρκεια αυτού του γεγονότος tickαυτό που πρόκειται να επιτύχουμε είναι να βεβαιωθούμε ότι τοπουλί καθώς και το αυγό - πουλίθα βρίσκεται εντός των ορίων του παιχνιδιού.

5.6.2 Τοποθέτηση και κίνηση του πουλιού εντός ορίων του παιχνιδιού

Αρχικά μιλώντας πρώταγια το πουλί (γραμμές 245 έως 263) θα πάμε να σιγουρευτούμε ότι το enemyXPos είναι μικρότερο από το πλάτος και μεγαλύτερο από το μηδέν στον άξονα τουΧγια να είμαστεμέσα στο παιχνίδι – καμβά (γραμμές 245 έως και 252).Έτσι ώστε αν τοπουλίκινείταιστην οθόνη(δηλαδήenemyXPos η θέση τουπουλιού στην οθόνη) πρόκειται να προσθέσουμε το στοιχείο της ταχύτητας της (δηλαδή το enemyXSpeed) έτσιώστε η ταχύτητα να καθορίζει πόσο γρήγορα θα πάει (γραμμή 247). Ύστεραπηγαίνουμε και δηλώνουμε τώρα ότι αν είναι εκτός οθόνης θα πολλαπλασιάσουμε την ταχύτητα με (-1) που πρόκειται να αντιστρέψει τη φορά και την κατεύθυνση της κίνησης τουπουλιού (γραμμή 250) και στη συνέχεια θα το ξαναμετακινήσουμε και πάλι (γραμμή 251).Έτσι θα κάνουμεακριβώς το ίδιο πράγμα και για τον άξονα y (γραμμή 253 έως 260) αφού σιγουρευτούμε ότι έχουμε ορίσει το enemyYPos να είναι μικρότερο από το πλάτος και μεγαλύτερο από το μηδέν στον άξονα του

(γραμμή 253. Στην συνέχεια στη γραμμή 255 αφού το πουλί κινείται στην οθόνη (δηλαδή enemyYPos η θέση του πουλιού στην οθόνη) πρόκειται να ξαναπροσθέσουμε το στοιχείο της ταχύτητας της (δηλαδή το enemyYSpeed) έτσι ώστε η ταχύτητα να καθορίζει πόσο γρήγορα θα πάει. Τέλος πηγαίνουμε και δηλώνουμε ξανά ότι αν είναι εκτός του καμβά μας θα πολλαπλασιάσουμε την ταχύτητα με (-1) που πρόκειται να αντιστρέψει τη φορά και την κατεύθυνση της κίνησης του πουλιού, και στη συνέχεια θα το ξαναμετακινήσουμε και πάλι (γραμμές 258 και 259 αντίστοιχα).

Τέλος μετακινούμε τοπουλίχρησιμοποιώντας τις θέσεις x και y του animation που ορίσαμε σε αυτήνπροηγουμένως και έπειτα θέτουμε τη θέση τουenemy η οποίακαθορίζεταιπλέον από την ταχύτητα του και στη συνέχεια το θέτουμε στο animation που είναι το πουλί μας χρησιμοποιώντας τις παραμέτρους x και y(Βλέπεγραμμές 262 και 263).

5.6.3 Τοποθέτηση και κίνηση του αυγού πουλιού εντός ορίων του παιχνιδιού

Περνώντας λοιπόν στην δεύτερη λειτουργεία του tickevent η οποίακάνει λόγο για το ότι πρέπει το αυγό – πουλί πρέπει και αυτό να βρίσκεται εντός των ορίων του παιχνιδιού μας και να κουνιέται πέρα δώθε χωρίς να χάνεται από το οπτικό μας πεδίο (γραμμές 223 έως 241). Αυτό λοιπόν το πετυχαίνουμε ακριβώς με τον ίδιο τρόπο όπως και κάναμε για το πουλί.

Έτσι και εδώ με την σειρά μας πρέπει να σιγουρευτούμε ότι το enemyXPos2 για τον άξονα του X και το enemyYPos2 για τον άξονα του y είναι μικρότερο από το πλάτος και μεγαλύτερο από το μηδέν για να είμαστε μέσα στο παιχνίδι (γραμμές 223 και 231 αντίστοιχα).Τώρα αν το αυγό - πουλί κινείται στην οθόνη(δηλαδή enemyXPos2 και enemyYPos2 η θέση του πουλιού στην οθόνη στον άξονα του X και Yαντίστοιχα) πρόκειται να προσθέσουμε το στοιχείο της ταχύτητας της (δηλαδή το enemyXSpeed2 και το enemyYSpeed2) έτσι ώστε η ταχύτητα να καθορίζει πόσο γρήγορα θα πάει (γραμμή 225 για τον άξονα του X και γραμμή 233 για τον άξονα του Y). Ύστερα πηγαίνουμε και δηλώνουμε τώρα ότι αν είναι εκτός οθόνης (δηλαδή στην περίπτωση του else) θα πολλαπλασιάσουμε την ταχύτητα ,και στον άξονα του X και στον άξονα του Y (enemyXSpeed2,enemyYSpeed2),με (-1) που πρόκειται να αντιστρέψει τη φορά και την κατεύθυνση της κίνησης του πουλιού (γραμμές 228 και 236

αντίστοιχα) και στη συνέχεια θα το ξαναμετακινήσουμε και πάλι, και στον άξονα του X και στον άξονα του Y, (γραμμές 229 και 237).

Τέλος μετακινούμε το αυγό - πουλί χρησιμοποιώντας τις θέσεις x και y του animation2 που ορίσαμε σε αυτήν προηγουμένως και έπειτα θέτουμε τη θέση του enemy η οποία καθορίζεται πλέον από την ταχύτητα του και στη συνέχεια το θέτουμε στο animation2 που είναι το αυγό - πουλί μας χρησιμοποιώντας τις παραμέτρους x και y (γραμμές 240 και 241).

```
160     // Προσθηκη ticker
161     createjs.Ticker.setFPS(15);
162     createjs.Ticker.addEventListener('tick', stage);
163     createjs.Ticker.addEventListener('tick', tickEvent);
```

Εικόνα 22

```
219 function tickEvent()
220 {
221
222     //Το εggπουλί πρεπει να βρίσκεται εντός των ορίων του παιχνιδιού και να κουνιεται περα δωθε
223     if(enemyXPos2 < WIDTH && enemyXPos2 > 0)
224     {
225         enemyXPos2 += enemyXSpeed2;
226     } else
227     {
228         enemyXSpeed2 = enemyXSpeed2 * (-1);
229         enemyXPos2 += enemyXSpeed2;
230     }
231     if(enemyYPos2 < HEIGHT && enemyYPos2 > 0)
232     {
233         enemyYPos2 += enemyYSpeed2;
234     } else
235     {
236         enemyYSpeed2 = enemyYSpeed2 * (-1);
237         enemyYPos2 += enemyYSpeed2;
238     }
239
240     animation2.x = enemyXPos2;
241     animation2.y = enemyYPos2;
242
243
244     //Το πουλί πρεπει να βρίσκεται εντός των ορίων του παιχνιδιού και να κουνιεται περα δωθε
245     if(enemyXPos < WIDTH && enemyXPos > 0)
246     {
247         enemyXPos += enemyXSpeed;
248     } else
249     {
250         enemyXSpeed = enemyXSpeed * (-1);
251         enemyXPos += enemyXSpeed;
252     }
253     if(enemyYPos < HEIGHT && enemyYPos > 0)
254     {
255         enemyYPos += enemyYSpeed;
256     } else
257     {
258         enemyYSpeed = enemyYSpeed * (-1);
259         enemyYPos += enemyYSpeed;
260     }
261
262     animation.x = enemyXPos;
263     animation.y = enemyYPos;
264 }
```

Εικόνα 23

5.7HandleMouseDown(event)

Τώραπάμε να δούμε ,μια από τις μεγαλύτερεςενότητεςόσοναναφορά το θέμα του κώδικα,την έκτηενότητα (γραμμές 268 έως 353, Εικόνα 25 και 26),η οποίαέχειάμεσησχέση με την δεύτερηενότητα και ποιο συγκεκριμένα στην γραμμή 166. Σε αυτήν την ενότητα θα ασχοληθούμε με το αν όντως χτυπήσαμε ή δεν χτυπήσαμε το πουλί ή το αυγό - πουλί, έχουμε να φτιάξουμε το γεγονός ότι πρέπει και τα δύο πουλιά να κουνιούνται σταδιακά όλο και ποιο γρηγορά καθώς πυροβολάμε με την υποτιθέμενη καραμπίνα μας, δηλαδή θα ασχοληθούμε με την δυσκολία του παιχνιδιού, τον ήχο που θα παίζει όταν πυροβολάμε ,θα φροντίσουμε για το σκορ και στη συνέχεια τι γίνεται αν χτυπήσουμε το πουλί ή το αυγό - πουλί(δηλαδή να αναγεννά μια νέα έκδοση πουλιού ή αυγού - πουλιού) ή δεν χτυπήσουμε κανένα από τα δύο.

5.7.1 Σετάρισμα των Events

Αρχικά θα δημιουργήσουμε την εντολή στη γραμμή 166Εικόνα 27 μετά το φόρτωμα του παιχνιδιού έτσι ώστε το onmousedown θα καλέσει handlemousedown. Αυτή είναι η λειτουργία επανάκλησης(callbackfunction) ή αλλιώς το event που είναι το κλειδί για να έχει ο χρήστης αποτελεσματικά τον έλεγχο του παιχνιδιού επειδή καθώς κουνάμε το ποντίκι γύρω και γύρω όταν παίζουμε το παιχνίδι ή όταν πατάμε για να πυροβολήσουμε θα καλέσει onmousedown για να ενημερώσουμε το παιχνίδι κάτι που έχουμε ήδη κάνει για να δημιουργήσουμε το πουλί καθώς και το αυγό – πουλί.

5.7.2 Απεικόνιση Στόχου – Παίξιμο ήχου Καραμπίνας

Έτσι το πρώτο πράγμα που κάνουμε στο handlemousedown(event) είναι όταν το ποντίκι κινείται, ο στόχος την στιγμή που πυροβολάμε να είναι ακριβώςεκεί που είναι και το ποντίκιδηλαδή να μετακινείτεμαζί του χωρίς όμως να φαίνεται παρά μόνο όταν πυροβολούμε. Έτσι τώρα θα δημιουργήσουμε το stoxos(γραμμή 272)με το νέο παράδειγμα μας createjs.bitmapκαλώντας το stoxos από την queue και στη συνέχεια το προσθέτουμε στο Childκαι κατά συνέπεια στο stage με το AddChild (γραμμή 275)αφούπρώταστις γραμμές 273 και 274έχουμε το event.client για το X και για το Y τα οποίακαθορίζουν την θέση του

ποντικιούαφαιρώντας τα με 45 (η διάστασητου στόχου στον X και Υάξονα έτσι όπως τον έχουμε δημιουργήσει από το Krita) έτσι ώστε το ποντίκι να εμφανίζεται στη μέση του stoxos και στη συνέχεια το αντιστοιχίζουμε στο x και y του stoxos, έτσι ώστε να μετακινούμε το ποντίκι παρέα με τον στόχο μας.

Στην συνέχεια θα παίξουμε τον πυροβολισμό ήχου, που είναι αυτό το είδος του ήχου καραμπίνα που ακούμε όταν παίζουμε το παιχνίδι κάθε φορά που πυροβολούμε(γραμμή 279).

5.7.3 Αύξηση ταχύτητας πουλιού επί τις %, υπολογισμός θέσης της βολής και απόστασης X και Y

Έτσιστις γραμμές 282 και 283 για το πουλί και στις 284 και 285 για το αυγό - πουλί, κάθε φορά που πυροβολούμεανεξάρτητα ,αυξάνουμεπερίπου 5% του enemyXSpeed, enemyXSpeed2 και6% στο enemyYSpeed, enemyYSpeed2 που είναι ακριβώς για να το κάνει λίγο διαφορετικό στον άξονα x και y. Τα ποσοστά προκύπτουν από τα αρχικά variables που είχαμε ορίσει στην αρχή του κώδικα με το 5% να ισούται με το 1.05 και το 6% να ισούται με το 1.06.Τώρα χρησιμοποιώντας το handleMouseDown(event)μαζί με το clientX και το clientYπου έχουμετοποθετήσειμέσα στο Math.roundθέλουμε να υπολογίσουμε το πού πυροβολήσαμεκαι ποιο από τα δυο πουλιά και να μάθουμε ποια είναι η θέση στον άξοναx και στον y στην οποία πραγματοποιήθηκε η βολή. Θα μας δώσειδηλαδή τις ακριβείςx,yπαραμέτρους(Βλέπεγραμμές 289έως 294).Ύστεραυπολογίζουμε την απόσταση μεταξύ τουx και yχρησιμοποιώντας τις αντίστοιχεςαπόλυτεςτιμέςτων παραμέτρων που έχουν προκύψει προηγουμένως από τις γραμμές 289 έως 294 δηλαδή το ShotX με το SpriteX και το ShotY με το SpriteY για το πουλί καθώς και το ShotX με το SpriteX2 και το ShotY με το SpriteY2 για το αυγό – πουλί (Βλέπε γραμμές 297 έως 300).

5.7.4 Καθορισμός σωστής βολής εκτός των φτερών στο αυγό πουλί - πουλί

Στη συνέχεια προσθέτουμε δύο ακόμα variables και τα ισοδυναμούμε με false (δηλαδή ψευδή) τα οποία τα ονομάζουμε Hit1, Hit2 (γραμμή 304).Στην ουσία η συγκεκριμένη αυτή μικρή εντολή είναι ίσως από τις σημαντικότερες διότι μας καθοδηγεί και μας πηγαίνει ανάλογα

με το αν η βολή μας είναιεπιτυχής η όχι,σε διαφορετικό μέρος του κώδικα. Δηλαδή, όταν το Hit1 ή το Hit2 είναι true (δηλαδήέχουμεπετύχει ένα από τα δύο πουλιά) μας πηγαίνει αυτόματα στις γραμμές 305 έως 325 αν έχουμε πετύχει το πουλί ή στις γραμμές 326 έως 345 αν έχουμε πετύχει το αυγό – πουλί. Στην περίπτωση τώρα που το Hit1 και το Hit2 είναι falseδηλαδή δεν έχουμε πετύχεικανένα από τα δύο πουλιά,τότεπερνάμε αυτόματα στις γραμμές 346 έως 352 όπου και ασχολούμαστε με την περίπτωση πλέον της αστοχίας.

Αρχικά με βάσητις προηγούμενεςαπόλυτεςτιμέςπερνάμε στην γραμμή 305 και 326 οι οποίες ισχύουν ήαλλιώς είναι αληθής όταν οι απόλυτεςτιμές που υπολογίσαμε στις γραμμές 297 και 298 για το πουλί και στις γραμμές 299 και 300 για το αυγό - πουλίείναι μικρότερες του 30 και του 60 για το πουλί και μικρότερες του 35 και του 60 για το αυγό - πουλίαντίστοιχα για X και Y του κάθε πουλιού.Αυτό,με βάσητιςφωτογραφίες - Spritesheets που έχουμεφτιάξειμέσω του Krita και για τα δυο πουλιά,σημαίνει ότι δεκτό είναι ένα χτύπημαμόνο όταν ακουμπήσει ο στόχος και πυροβολήσουμεοπουδήποτεταπουλιάεκτός από το κεφάλι.

Στην συνέχεια όταν πετύχουμετο πουλί ή το αυγό - πουλί πρέπει να τοκάνουμεremove από το stage με την εντολήremoveChild(animation – animation 2) (γραμμές 309 και 330 αντίστοιχα).Έπειτα στιςγραμμές 310 και 331καλούμε τιςεντολέςbirdDeath - Eggbirddeath στιςοποίεςαναφερθήκαμεκυρίως στην τέταρτηλειτουργία του κώδικα. Επιστρέφονταςξανά στην περίπτωση που πετύχουμετο πουλίή το αυγό – πουλί και ποιο συγκεκριμένα στιςγραμμές311 και 332 οιοποίες με την εντολήscore+=100 μας βοηθάνε κάθε φορά που πετυχαίνουμεένα από τα δύο πουλιά να κερδίζουμε 100 πόντους στο scoretextμας. Σεσυνδυασμόπάντα με τιςγραμμές 312 και 333στην οποίαενημερώνουμε το scoreText, μας χρησιμοποιώντας την εντολήscoreText.text που περιέχει το κείμενοπόντων που θέλουμε να αναγράφετε όταν παίζουμε το παιχνίδι, και το Score.toString.

Τέλος στις γραμμές 313 και 334, οι οποίες είναι ακριβώςίδιες,παίζουμε έναν ήχο που πραγματοποιείται όταν πεθαίνειτο πουλί. Επειδήτώρα κάθε παιχνίδιπρέπει να έχει και ένα βαθμόδυσκολίας κάθε φορά που πετυχαίνουμετο πουλί με βάση τις γραμμές 316 και 317είτε το αυγό – πουλί στις γραμμές 337 και 338,ο βαθμόςδυσκολίας τους ανεβαίνει κατά 25% στον άξονα του Y και κατά 30% στον άξονα του X που ισοδυναμείτε με τοenemyYSpeed=enemyYSpeed*PerCentDiffXή
enemyYSpeed2=enemyYSpeed*PerCentDiffXγια το Y του πουλιού ή το Y του αυγού -

πουλιούαντίστοιχα και με το enemyXSpeed=enemyXSpeed*PerCentDiffYή enemyXSpeed2=enemyXSpeed*PerCentDiffYγια το X του πουλιού ή για το X του αυγού – πουλιού αντίστοιχα.

Τέλος θα χρειαστεί να ρυθμίσουμε τον χρόνο που κάνει να ξαναβγεί το πουλί ή το αυγό - πουλί από την στιγμή που το πυροβολούμε. Όπως μπορούμε να δούμε στις γραμμές 321 και 342 (οι οποίες είναι ακριβώς ίδιες και στις δυο περιπτώσεις και Hit1=true και Hit2=true) περιμένουμε ανάμεσα στο 1 δευτερόλεπτο και τα 3.000 χιλιοστά του δευτερολέπτου(3 δευτερόλεπτα δηλαδή) μέχρι να ξαναβγεί το επόμενο ίδιο πουλί και λέμε από το 1 έως τα 3 δευτερόλεπτα γιατί το έχουμε ρυθμίσει κάθε φορά να είναι τυχαίος ο χρόνος που θα κάνει να ξαναεμφανιστεί από την εντολή Math.random που προσθέσαμε στην Math.floor που ισούται με την timeToCreate.

Έπειτα στην γραμμή 322 στην εντολή setTimeout έχουμε βάλει τις εντολές createEnemy και timeToCreate και στην γραμμή 343 στην εντολή setTimeout έχουμε βάλει τις εντολές createEnemy2 και timeToCreate. Αφού καθοριστεί τυχαία ο χρόνος μέσα στην εντολή setTimeout εκτελεί την εντολή createEnemy ή createEnemy2 ,ανάλογα για ποιο πουλί γίνονται οι ενέργειες, δημιουργώντας ένα νέο μετά από μια χρονική περίοδο φυσικά λόγο του timeToCreate.

```
function handleMouseDown(event)
{

    //απεικόνιση στοχου
    stoxos = new createjs.Bitmap(queue.getResult("stoxos"));
    stoxos.x = event.clientX-45;
    stoxos.y = event.clientY-45;
    stage.addChild(stoxos);
    createjs.Tween.get(stoxos).to({alpha: 0},1000);

    //παιξιμο ηχου 'shot' καραμπινας
    createjs.Sound.play("shot");

    //αυξηση ταχυτητας πουλιου επι τις %
    enemyXSpeed *= PerCentSpeedX;
    enemyYSpeed *= PerCentSpeedY;
    enemyXSpeed2 *= PerCentSpeedX;
    enemyYSpeed2 *= PerCentSpeedY;


    //υπολογισμος της θεσης οπου πραγματοποιηθηκε η βολη
    var shotX = Math.round(event.clientX);
    var shotY = Math.round(event.clientY);
    var spriteX = Math.round(animation.x);
    var spriteY = Math.round(animation.y);
    var spriteX2 = Math.round(animation2.x);
    var spriteY2 = Math.round(animation2.y);

   // Υπολογίσμος της απόστασης X και Y χρησιμοποιώντας απόλυτη τιμή
    var distX = Math.abs(shotX - spriteX);
    var distY = Math.abs(shotY - spriteY);
    var distX2 = Math.abs(shotX - spriteX2);
    var distY2 = Math.abs(shotY - spriteY2);


    // Χτυπημα οπουδηποτε στο πουλι εκτος των φτερων θερωρηται χτυπημα
    var Hit1=false, Hit2=false;
    if( distX < CoSPx && distY < CoSPy )
    {
        Hit1=true;
        //Σε περιπτωση χτυπηματος πουλιου και κερδισμα ποντων
        stage.removeChild(animation);
        birdDeath();
        score += scoreup;
        scoreText.text = "POINTS: " + score.toString();
        createjs.Sound.play("deathSound");

        //Δυσκολια του παιχνιδιου
        enemyYSpeed *= PerCentDiffX;
        enemyXSpeed *= PerCentDiffY;


        //Δημιουργιας νεου πουλιου
        var timeToCreate = Math.floor((Math.random()*3000)+1);
        setTimeout(createEnemy,timeToCreate);


    }
```

Εικόνα 24

```
326     if ( distX2 < CoSPx2 && distY2 < CoSPy2 )
327     {
328         Hit2=true;
329         //Σε περιπτωση χτυπηματος πουλιου και κερδισμα ποντων
330         stage.removeChild(animation2);
331         EggBirdDeath();
332         score += 100;
333         scoreText.text = "POINTS: " + score.toString();
334         createjs.Sound.play("deathSound");
335 
336         //Δυσκολια του παιχνιδιου
337         enemyYSpeed2 *= PerCentDiffX;
338         enemyXSpeed2 *= PerCentDiffY;
339 
340 
341         //Δημιουργιας νεου πουλιου
342         var timeToCreate = Math.floor((Math.random()*3000)+1);
343         setTimeout(createEnemy2,timeToCreate);
344 
345     }
346     if (Hit1 == false && Hit2 == false)
347     {
348         //Σε περιπτωση αστοχιας
349         score -= miss;
350         scoreText.text = "POINTS: " + score.toString();
351 
352      }
353 }
```

Εικόνα 25

```
165     // Σεταρισμα των events αφου φορτωθει το παιχνιδι
166     window.onmousedown = handleMouseDown;
```

Εικόνα 26

5.7.5 Περίπτωση αστοχίας

Στην περίπτωση όπου αστοχήσουμε λοιπόνπερνάμε αυτόματα, παραλείποντας την πρώτη περίπτωση του true, στιςγραμμές346έως 352 (Εικόνα 28).Στην γραμμή 346 ορίζουμε την συνθήκηHit1==false&&Hit2==false που σημαίνειαυτόματα ότι έχουμεαστοχήσει με αποτέλεσμα να μας πηγαίνειαμέσως μετά στην γραμμή 349 όπου καιέχουμε την εντολήscore-=miss η οποίααπλά μας αφαιρεί 10 πόντους(γιατίέτσιακριβώς έχουμε ορίσει την λέξηmiss από τα αρχικά μας variables)κάθε φορά που πατάμε κλικ και αστοχούμε και στην συνέχεια στηνγραμμή 350ενημερώνουμε το scoreText κάθε φορά που συμβαίνει αυτό όπως κάναμεφυσικά και στηνπερίπτωση ευστοχίας.

```
346     if (Hit1 == false && Hit2 == false)
347     {
348         //Σε περιπτωση αστοχιας
349         score -= miss;
350         scoreText.text = "POINTS: " + score.toString();
351 
352      }
```

Εικόνα 27

5.8UpdateTime

Τέλοςπάμε να δούμε και το τελευταίο κομμάτι του κώδικατο έβδομο δηλαδή το οποίο είναι εξίσουσημαντικό για την σωστήλειτουργία του παιχνιδιού μας από την γραμμή 356 μέχρι την γραμμή 375(Εικόνα 29) στις οποίες θα ασχοληθούμε με το τι κάνει η updateTimeεντολή και τι εμπεριέχονταιμέσα σε αυτήν.

5.8.1 Καθαρισμός οθόνης και τέλος του παιχνιδιού

Αρχικά στην γραμμή 358ορίζουμεανάπόσαδευτερόλεπτα(στην περίπτωση μας ανά 1)θα τρέχει και θα προσθέτει το χρονόμετρο μας στην πάνωδεξιάμεριά – γωνία της οθόνηςμας. Στην γραμμή 359ορίζουμε την εντολήgameTime όταν είναι μεγαλύτερο από το 60 δηλαδή το EndGameInSecondsνα περνάει αυτόματα στις γραμμές 362έως 369 (δηλαδήεπισημοποιεί το τέλος του παιχνιδιού) και όταν είναι μικρότερο(δηλαδή κατά την διάρκεια του παιχνιδιού) να πηγαίνει στις γραμμές 370 έως 374.

Τώραεξετάζοντας τηνπερίπτωση που είναι μεγαλύτεροπερνάμε στην γραμμή 362 η οποία μας δείχνει το κείμενο που φαίνεται όταν τελειώνει το παιχνίδι.Στις γραμμέςτώρα 363-364,365 και 366κάνουμεremovetoanimation και το animation2 (δηλαδή το πουλί και το αυγό - πουλί),τον στόχοκαι σταματάει τον backgroundήχο που ακούγαμε όταν παίζαμε το παιχνίδι αντίστοιχα. Στην 367μέσω της εντολήςcreatejs.sound.playδημιουργούμε - παίζουμε έναν ήχο(gameOverSound) που έχουμεαποθήκευση στον φάκελοassets που σημάνει το τέλος του παιχνιδιού. Τέλος στην γραμμή 368απλάκαθαρίζουμεδηλαδήαφαιρούμε το χρονόμετρο να μην φαίνεταιπλέον.

5.8.2 Συνδυασμός tick ήχων με την αύξηση δευτερολέπτων κατά 1

Έπειταπερνάμε στην δεύτερηπερίπτωση μας στην γραμμή 370 και κατά συνέπεια στην γραμμή 372φυσικά στην οποίααπλάανεβάζουμε κατά 1 δευτερόλεπτο το χρονόμετρομας και στην 373 κάθε φορά που προσθέτουμε 1 δευτερόλεπτο να ακούγεται - παίζεται και ο “tick”

ήχος, που κάνει το χρονόμετρο καθώς ανεβαίνει κατά ένα, τον οποίοεπίσης έχουμεμέσα στον φάκελοassets.

```
356  function updateTime()
357  {
358      gameTime += TimePerSecond;
359      if(gameTime > EndGameInSeconds)
360      {
361          //Τελος του παιχνιδιου και καθαρισμο τα παντα απο την οθονη
362          timerText.text = "LOOOSER ";
363          stage.removeChild(animation);
364          stage.removeChild(animation2);
365          stage.removeChild(stoxos);
366          createjs.Sound.removeSound("background");
367          var si =createjs.Sound.play("gameOverSound");
368          clearInterval(gameTimer);
369      }
370      else
371      {
372          timerText.text = "Time: " + gameTime
373      createjs.Sound.play("tick");
374      }
375  }
```

Εικόνα 28

Βιβλιογραφία

- Πρόγραμμα Krita. [online] Διαθέσιμο από: <https://reviews.financesonline.com/p/krita/> [πρόσβαση 10/08/2018].

- JavaScript. [online] Διαθέσιμο από: <https://students.cs.unipi.gr/~xalgra/html5_unit0_js.pdf> [πρόσβαση 10/08/2018].

- Functions - Λειτουργίες. [online] Διαθέσιμο από: <https://docs.python.org/3.1/library/functions.html> [πρόσβαση 10/08/2018].

- Πρόγραμμα Audacity. [online] Διαθέσιμο από: <https://www.audacityteam.org/about/> [πρόσβαση 10/08/2018].

- "Why did we build Visual Studio Code?". [online] Διαθέσιμοαπό: <https://code.visualstudio.com/docs/editor/whyvscode> [πρόσβαση 10/08/2018

- "What is Visual Studio Code?". [online] Διαθέσιμο από: <https://en.wikipedia.org/wiki/Visual_Studio_Code> [πρόσβαση 10/08/2018].

- JavaScript Libraries and Tools. [online] Διαθέσιμοαπό: <https://www.sitepoint.com/top-javascript-frameworks-libraries-tools-use/> [πρόσβαση 10/08/2018].

- JavaScript. [online] Διαθέσιμο από: <https://el.wikipedia.org/wiki/JavaScript> [πρόσβαση 10/08/2018].

- CreateJS Libraries. [online] Διαθέσιμοαπό: <https://createjs.com/downloads> [πρόσβαση 10/08/2018].

- DOT.JS. [online] Διαθέσιμο από: <http://olado.github.io/doT/index.html> [πρόσβαση 10/08/2018].

- normalize.css. [online] Διαθέσιμοαπό: <https://github.com/necolas/normalize.css> [πρόσβαση 10/08/2018].

- About normalize.css. [online] Διαθέσιμοαπό: <http://nicolasgallagher.com/about-normalize-css/> [πρόσβαση 10/08/2018].

- LoadQueue Class. [online] Διαθέσιμοαπό: <https://createjs.com/docs/preloadjs/classes/LoadQueue.html> [πρόσβαση 10/08/2018].

- Stage Class. [online] Διαθέσιμοαπό: <https://createjs.com/docs/easeljs/classes/Stage.html#method_addChild> [πρόσβαση 10/08/2018].

- SpriteSheet.JS. [online] Διαθέσιμο από: <https://github.com/krzysztof-o/spritesheet.js> [πρόσβαση 10/08/2018].

- GotoAndPlay. [online] Διαθέσιμο από: <https://www.dummies.com/software/adobe/flash/how-to-use-gotoandplay-and-gotoandstop-in-flash-cs5-actionscript/> [πρόσβαση 10/08/2018].