Τ.Ε.Ι ΚΡΗΤΗΣ
ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ
ΤΜΗΜΑ ΕΦΑΡΜΟΣΜΕΝΗΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΠΟΛΥΜΕΣΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΑΝΑΠΤΥΞΗ ΠΑΙΧΝΙΔΙΟΥ ΤΥΠΟΥ «SPACE SHOOT'EM UP» ΓΙΑ PDA

Σπουδαστής: Σπύρογλου Δημήτριος

Εισηγητής: Παχουλάκης Ιωάννης

Πρόλογος

Σκοπός της παρούσης πτυχιακής εργασίας είναι η ανάπτυξη ενός παιχνιδιού τύπου “Space shoot ‘em up” για PDA χρησιμοποιώντας παράλληλα τα σωστά εργαλεία και τις κατάλληλες μεθόδους προγραμματισμού έτσι ώστε η κατασκευή του να στηριχτεί σε μια ευέλικτη, με υψηλές δυνατότητες παραμετροποίησης, «μηχανής» παιχνιδιού. Αποτέλεσμα μιας τέτοιας προσέγγισης είναι η γρήγορη και εύκολη παραμετροποίηση του κώδικα, ώστε διαφορετικές εκδόσεις του παιχνιδιού να μπορούν να κατασκευαστούν εύκολα και γρήγορα χωρίς να χρειάζονται περαιτέρω αλλαγές στα βασικά του κομμάτια.

Αρχικά, στο πρώτο κεφάλαιο θα γίνει μια εισαγωγική αναφορά στο περιβάλλον υλοποίησης, στα εργαλεία που επιλέχθηκαν για την ανάπτυξη και την υλοποίηση του όλου project, μια εισαγωγή στο παιχνίδι και μερικές οδηγίες για το πως παίζεται. Στη συνέχεια, στο δεύτερο κεφάλαιο θα αναλυθεί η προγραμματιστική προσέγγιση και οι λεπτομέρειες κατασκευής της καρδιάς της εφαρμογής, της «μηχανής» παιχνιδιού. Στο τελευταίο κεφάλαιο θα παρουσιαστούν ενδεικτικοί τρόποι με τους οποίους διαφορετικές εκδόσεις του παιχνιδιού μπορούν να δημιουργηθούν με ευκολία από τον προγραμματιστή, ακόμα και από τον χρήστη.

Περιεχόμενα

1.Εισαγωγή

Η ανάπτυξη του παιχνιδιού τύπου «Space shoot' em up» για PDA έγινε σε περιβάλλον VB .NET Compact Framework, χρησιμοποιώντας το Visual Studio. Η επιλογή της VB .NET έγινε επειδή το παιχνίδι έχει πολλά αντικειμενοστραφή στοιχεία και η συγκεκριμένη γλώσσα εξυπηρετούσε την ανάπτυξη, αλλά και λόγω των εύχρηστων API's που διαθέτει και συγκεκριμένα του GDI+ το οποίο ασχολείται με την σχεδίαση των γραφικών.

1.1 Εισαγωγή στα PDA

Ένα PDA (Personal Digital Assistant – Προσωπικός ψηφιακός βοηθός), γνωστό και ως υπολογιστής χειρός ή παλάμης, είναι ένας μικρού μεγέθους φορητός υπολογιστής, με διαστάσεις που του επιτρέπουν να χωράει στην τσέπη ή την παλάμη του χεριού. Αποτελείται από μια μικρή οθόνη, συνήθως αφής και από μερικά πλήκτρα ελέγχου ή πολλές φορές ακόμα και από ένα πλήρες μίνι-πληκτρολόγιο. Έχουν δυνατότητα σύνδεσης στο internet, ενώ μπορούν να χρησιμοποιηθούν και ως κινητά τηλέφωνα αλλά και ως φορητές συσκευές αναπαραγωγής ήχου.

Ένα δημοφιλές λειτουργικό σύστημα για PDA είναι τα Windows Mobile, ένα λειτουργικό που αναπτύχθηκε από την Microsoft. Πολλές εταιρείες κατασκευής PDA χρησιμοποιούν λειτουργικό Windows Mobile όπως οι Toshiba, Airis, Asus, Cyberbank, Dell, Fujitsu Siemens, HP, Motorola κ.α. Τα PDA αυτά που βασίζονται στα Windows Mobile αναφέρονται και ως Pocket PC (PPC). Τα Windows Mobile δίνουν στον χρήστη ενός PDA τη δυνατότητα χρήσης της συσκευής του σε ένα εύχρηστο γραφικό περιβάλλον, στο οποίο μπορεί να τρέξει εφαρμογές γραφείου, όπως λογιστικά φύλλα, organizer και πολλά άλλα. Μια δημοφιλής κατηγορία εφαρμογών είναι και αυτή των ηλεκτρονικών παιχνιδιών.

Ένα HP iPaq Pocket PC

1.2 VB.NET και .NET Compact Framework

Η Visual Basic .NET (VB.NET) είναι μια αντικειμενοστραφής γλώσσα προγραμματισμού η οποία ουσιαστικά αποτελεί την εξέλιξη της Visual Basic (VB) της Microsoft. Η Visual Basic .NET ήταν η πρώτη πλήρως αντικειμενοστραφής έκδοση της Visual Basic και υποστηρίζει αντικειμενοστραφή στοιχεία όπως κληρονομικότητα και πολυμορφισμό. Υλοποιήθηκε πάνω στο .NET Framework το οποίο αποτελείται από μια πλούσια βιβλιοθήκη λειτουργιών που υλοποιούν λειτουργίες διεπαφής χρήστη-υπολογιστή, πρόσβασης δεδομένων, συνδεσιμότητα με βάσεις δεδομένων, κρυπτογράφησης, επικοινωνιών δικτύου, υποστήριξης πολυμέσων κ.α.

Το .NET Compact Framework είναι μια έκδοση του .NET Framework ειδικά σχεδιασμένη για συστήματα με περιορισμένους πόρους, όπως τα PDA. Κάποιες λειτουργίες από το .NET Framework χρησιμοποιούνται και στο .NET Compact Framework, ενώ υπάρχουν και κάποιες ειδικά τροποποιημένες ή σχεδιασμένες για να απαιτούν λιγότερους πόρους, έτσι ώστε να μπορούν να χρησιμοποιηθούν σε συστήματα όπως PDAs.

1.3 Microsoft Visual Studio

Το Microsoft Visual Studio είναι ένα ολοκληρωμένο περιβάλλον ανάπτυξης λογισμικού, κατασκευασμένο από την Microsoft. Χρησιμοποιείται για την ανάπτυξη διάφορων εφαρμογών για περιβάλλοντα Microsoft Windows, Windows Mobile, Windows CE, .NET Framework, .NET Compact Framework και Microsoft Silverlight. Το Microsoft Visual Studio υποστηρίζει πολλές γλώσσες προγραμματισμού όπως C/C++, VB.NET και άλλες. Για την εκτέλεση της PDA εφαρμογής του παιχνιδιού τύπου «Space shoot 'em up» , χρησιμοποιείται ένας emulator ο οποίος εξομοιώνει πλήρως μια συσκευή Pocket PC με λειτουργικό Windows Mobile 5.0. Συγκεκριμένα ο USA Windows Mobile 5.0 Pocket PC R2 Emulator.

USA Windows Mobile 5.0 Pocket PC R2 Emulator

1.4 GDI+

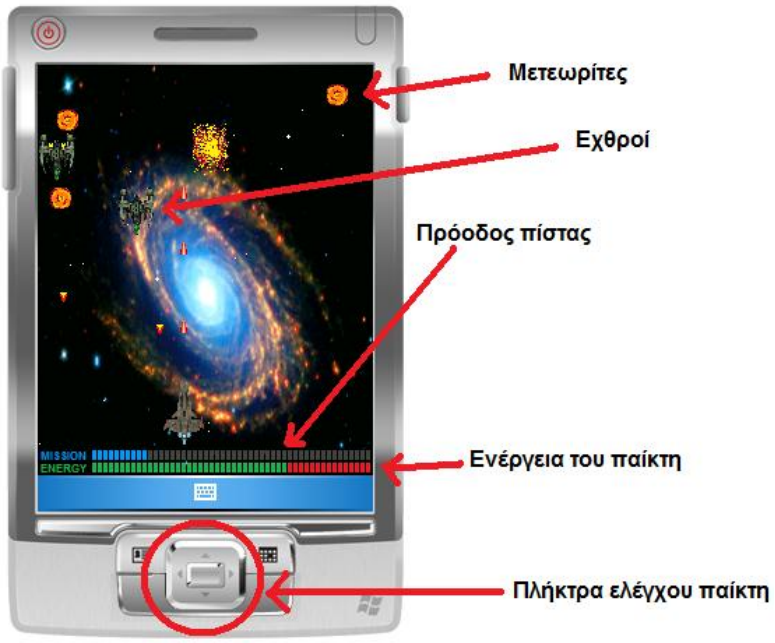
Το Graphics Device Interface (GDI) είναι μια διασύνδεση προγραμματισμού εφαρμογών (Application Programming Interface – API) η οποία υλοποιεί λειτουργίες σχεδίασης και ελέγχου γραφικών όπως γραμμές, καμπύλες, γραμματοσειρές, γραφικά κ.α.

Το GDI+ είναι μια εξέλιξη του GDI. Το GDI+ προσφέρει περισσότερες λειτουργίες σχεδίασης σε σχέση με το GDI, αλλά και υποστήριξης μεγαλύτερης γκάμας γραφικών format, όπως JPEG και PNG. Το GDI+ είναι μέρος του .NET Framework αλλά και του .NET Compact Framework και είναι διαθέσιμο στον προγραμματιστή μέσα από το *System.Drawing* namespace.

1.5 Παιχνίδι Space Shoot'em up – Οδηγίες χειρισμού

Τα παιχνίδια τύπου Space Shoot'em up αφορούν παιχνίδια στα οποία ο παίκτης χειρίζεται ένα διαστημόπλοιο, το οποίο συνήθως βρίσκεται σε κάποιο άκρο της οθόνης και αντιμετωπίζει εχθρικά διαστημόπλοια, εξωγήινους κλπ καθώς ταξιδεύει στο διάστημα. Είναι μια ιδιαίτερα δημοφιλής και διαχρονική κατηγορία ηλεκτρονικών παιχνιδιών.

Το παιχνίδι που υλοποιήθηκε στηρίζεται σε αυτή την συνταγή. Ο παίκτης ελέγχει ένα διαστημόπλοιο στο κάτω μέρος της οθόνης, το οποίο μπορεί να κινήσει με τα αντίστοιχα βελάκια του PDA ενώ μπορεί να πυροβολήσει χρησιμοποιώντας το μεσαίο πλήκτρο επιλογής. Το παιχνίδι χωρίζεται σε διάφορες πίστες, κάθε μια με διαφορετικό επίπεδο δυσκολίας και εχθρούς. Στόχος είναι να βγάλει εις πέρας όλες τις πίστες χωρίς να καταστραφεί το διαστημόπλοιό του.



2. Προγραμματισμός του παιχνιδιού

Σε αυτό το κεφάλαιο θα εξηγηθούν ενδεικτικά κομμάτια του κώδικα, ο ρόλος των μεταβλητών και ο τρόπος που επιδρούν οι διάφορες λειτουργίες στον αλγόριθμο που υλοποιεί το παιχνίδι. Πλήρες αντίγραφο του κώδικα είναι διαθέσιμο στο συνοδευτικό CD.

2.1 Περιγραφή της προγραμματιστικής προσέγγισης

Ο τρόπος σκέψης, σχεδίασης και υλοποίησης του κώδικα βασίστηκε και εκμεταλλεύτηκε σε πολύ μεγάλο βαθμό την κληρονομικότητα. Τα κυρίως αντικείμενα από τα οποία αποτελείται το παιχνίδι, όπως το διαστημόπλοιο του παίκτη, οι διάφοροι εχθροί, τα bonus αντικείμενα, τα laser μέχρι και τα background αστέρια, έχουν αρκετές ομοιότητες στη συμπεριφορά τους. Από παρόμοια χαρακτηριστικά, όπως η θέση τους X,Y στην οθόνη, μέχρι σε κοινές ιδιότητες, όπως η σχεδίασή τους στην οθόνη ή ο έλεγχος συγκρούσεων. Έτσι κρίθηκε αναγκαία η ομαδοποίησή τους και πιο συγκεκριμένα η δημιουργία βασικών κλάσεων στις οποίες θα υπάρχουν όλα αυτά τα κοινά χαρακτηριστικά και λειτουργίες.

2.2 Η GameObject και η GameObjectAnimated

Οι “μητρικές” κλάσεις όλων των αντικειμένων του παιχνιδιού. Η GameObject και η GameObjectAnimated είναι MustInherit κλάσεις, δηλαδή δεν μπορούν να χρησιμοποιηθούν αυτές καθεαυτές για την δημιουργία ενός αντικειμένου. Με άλλα λόγια ένα αντικείμενο δεν

μπορεί να είναι απλά τύπου GameObject ή GameObjectAnimated, αλλά θα πρέπει για τον τύπο του να χρησιμοποιηθεί μια άλλη κλάση η οποία θα κληρονομεί τα χαρακτηριστικά και τις λειτουργίες της από την GameObject ή την GameObjectAnimated. Πάνω στην GameObject και στην GameObjectAnimated στηρίζονται όλες οι κλάσεις που αφορούν τα κυρίως αντικείμενα του παιχνιδιού και πιο συγκεκριμένα οι κλάσεις:

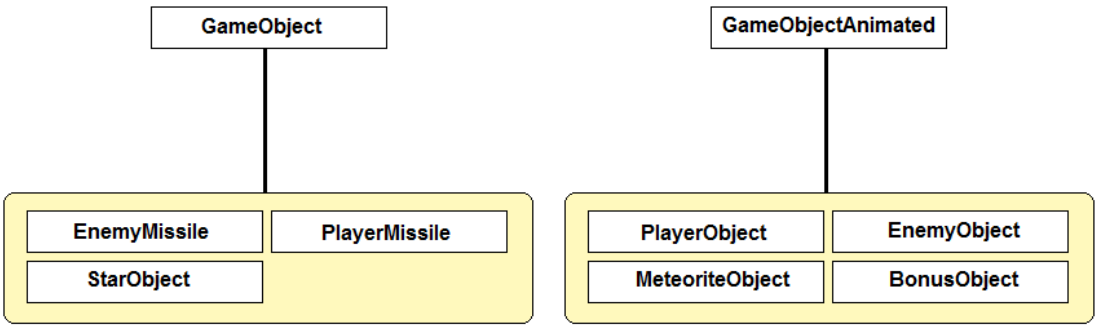
GameObject:

- StarObject
- PlayerMissile
- EnemyMissile

GameObjectAnimated:

- PlayerObject
- EnemyObject
- MeteoriteObject
- BonusObject

Κληρονομικότητα των κλάσεων



2.2.1 Περιγραφή της GameObject

Εξετάζοντας την GameObject μπορούμε να καταλάβουμε εύκολα την δομή των αντικειμένων του παιχνιδιού. Οι μεταβλητές που χρησιμοποιούνται αφορούν βασικά χαρακτηριστικά και λειτουργίες που μπορεί να έχει ένα αντικείμενο του παιχνιδιού.

ObjImage

Η εικόνα που χρησιμοποιεί το αντικείμενο.

MinX, MaxX, ObjX, ObjStepX

Βασικά χαρακτηριστικά που αφορούν την θέση και κίνηση του αντικειμένου στον X άξονα. Η MinX είναι η μικρότερη δυνατή θέση στην οποία μπορεί να βρεθεί ένα αντικείμενο στον άξονα X ενώ η MaxX είναι αντιστοίχως η μέγιστη. Η ObjX είναι μια βασική μεταβλητή η οποία αναφέρεται στην τρέχουσα θέση του αντικειμένου στον άξονα X. Η ObjStepX αναφέρεται στο πόσα pixel μετατοπίζεται κάθε φορά το αντικείμενο στον άξονα X κατά την κίνηση του σε αυτόν, είναι ουσιαστικά η οριζόντια ταχύτητα του αντικειμένου.

MinY, MaxY, ObjY, ObjStepY

Παρομοίως με τα αντίστοιχα X, μόνο που αφορούν τον άξονα Y.

ObjWidth, ObjHeight

Οι μεταβλητές αναφέρονται στο πλάτος και στο ύψος του αντικειμένου.

IsAlive

Ένα Boolean που αναφέρεται στο αν το αντικείμενο είναι “ενεργό” η όχι.

ObjSpeed, ObjMinSpeed, ObjMaxSpeed

Η ObjSpeed αναφέρεται στην ταχύτητα του αντικειμένου και παίρνει τιμές από ObjMinSpeed έως ObjMaxSpeed.

dstRect

Η dstRect χρησιμοποιείται από την ρουτίνα σχεδίασης του αντικειμένου στην οθόνη.

TransparencyAttr

Η TransparencyAttr χρησιμοποιείται από την ρουτίνα σχεδίασης του αντικειμένου στην οθόνη.

Constructor της GameObject

Ο Constructor παίρνει ως ορίσματα τις προαναφερθείσες μεταβλητές, ενώ αξίζει να σημειωθεί ότι στις αρχικοποιήσεις δίνει στην ObjSpeed μια τυχαία τιμή στο δοσμένο εύρος ObjMinSpeed, ObjMaxSpeed χρησιμοποιώντας την ObjRandomizer, ενώ επίσης δίνει την τιμή False στην Boolean IsAlive.

2.2.2 Λειτουργίες της κλάσης GameObject

PositionUpdate

Η PositionUpdate, όπως μαρτυράει η ονομασία της, ασχολείται με την ανανέωση της θέσης του αντικειμένου. Είναι MustOverride, που σημαίνει ότι η κλάση η οποία θα κληρονομήσει τα χαρακτηριστικά της GameObject θα πρέπει να υλοποιήσει την PositionUpdate. Καθώς κάθε αντικείμενο του παιχνιδιού έχει του δικούς του κανόνες κίνησης, η PositionUpdate για κάθε αντικείμενο πρέπει να είναι ξεχωριστή, υλοποιώντας τους εκάστοτε κανόνες κίνησης για την επίτευξη του επιθυμητού αποτελέσματος. Για παράδειγμα, ένα Laser κινείται μόνο στον άξονα Y, ενώ ένα διαστημόπλοιο και στους δύο άξονες, ένα εχθρικό διαστημόπλοιο έχει το δικό του μοντέλο κίνησης, ενώ το διαστημόπλοιο του παίκτη εξαρτάται από τις εντολές που δίνει ο χρήστης από το πληκτρολόγιο. Μπορεί όλα τα αντικείμενα να κινούνται διαφορετικά, όλα όμως κινούνται. Για αυτό τον λόγο όλα υλοποιούν το κάθε ένα ξεχωριστά την PositionUpdate.

BoundCheck

Η BoundCheck ασχολείται με το αν το αντικείμενο κινείται μέσα στα επιτρεπτά όρια. Η GameObject υλοποιεί ένα βασικό αλγόριθμο ελέγχου χρησιμοποιώντας τα βασικά στοιχεία που έχει στην διάθεσή της, τα MinX, MaxX και MinY, MaxY. Με λίγα λόγια περιορίζει το αντικείμενο μέσα στον διαθέσιμο χώρο κίνησής του. Είναι Overridable, που σημαίνει ότι μια κλάση μπορεί να αν θέλει να υλοποιήσει με δικούς της πιθανόν πιο πολύπλοκους κανόνες μια τέτοια λειτουργία.

```
    If ObjX >= MaxX Then ObjX = MaxX
    If ObjY >= MaxY Then ObjY = MaxY
    If ObjX <= MinX Then ObjX = MinX
    If ObjY <= MinY Then ObjY = MinY
```

DrawObject

Η DrawObject ασχολείται με την σχεδίαση του αντικειμένου στην οθόνη. Πιο σωστά με την σχεδίαση του αντικειμένου στο bitmap shared αντικείμενο bmpDoubleBuffer.

Πιο αναλυτικά:

```
Public Overridable Sub DrawObject()

      Dim g As Graphics = Graphics.FromImage(bmpDoubleBuffer)
      dstRect = New Rectangle(ObjX, ObjY, ObjImage.Width,
ObjImage.Height)
      TransparencyAttr.SetColorKey(Color.Red, Color.Red)
      g.DrawImage(ObjImage, dstRect, 0, 0, ObjImage.Width,
ObjImage.Height, GraphicsUnit.Pixel, TransparencyAttr)

End Sub
```

Εδώ χρησιμοποιούνται οι λειτουργίες του GDI+. Η g είναι τύπου Graphics και ως όρισμα δίνουμε το κυρίως bitmap αντικείμενο bmpDoubleBuffer. Το dstRect είναι ένα αντικείμενο

τύπου Rectangle και ως όρισμα του δίνουμε την θέση του αντικειμένου και τις διαστάσεις του. Το ορθογώνιο δηλαδή που καταλαμβάνει πάνω στην οθόνη. Η TransparencyAttr που είναι τύπου ImageAttributes, ρυθμίζεται έτσι ώστε το χρώμα που να μας δίνει διαφάνεια να είναι το κόκκινο. Στην συνέχεια με g.DrawImage γράφουμε πάνω στην g την εικόνα του αντικείμενού μας (ObjImage), στην συγκεκριμένη θέση και χώρο που πρέπει να έχει το αντικείμενο (το οποίο έχει οριστεί από το dstRect), την σχετική περιοχή του αντικειμένου που θέλουμε να σχεδιαστεί (στην προκειμένη περίπτωση όλο το αντικείμενο, δηλαδή σε XY από 0, 0 του αντικειμένου μέχρι το ObjImage.Width, ObjImage.Height), το measurement unit που είναι pixels και τέλος με το όρισμα TransparencyAttr επιτυγχάνουμε την "διαφάνεια" του αντικειμένου που έχει οριστεί το χρώμα κόκκινο.

2.2.3 Περιγραφή της GameObjectAnimated

Η GameObjectAnimated έχει κοινά χαρακτηριστικά με την GameObject με την διαφορά ότι απευθύνεται σε στοιχεία του παιχνιδιού τα οποία απαιτούν κάποιο animation. Για παράδειγμα, το διαστημόπλοιο του παίκτη δεν αποτελείται από μια μόνο εικόνα, όπως το laser του, αλλά έχει διάφορες εικόνες (frames) που δημιουργούν την αίσθηση της κινούμενης εικόνας και επιπλέον έχει και διάφορες καταστάσεις. Πιο συγκεκριμένα, έχει διαφορετική εικόνα όταν το διαστημόπλοιο έρχεται σε σύγκρουση με έναν εχθρό, αλλά και διαφορετική εικόνα όταν αυτό εκρήγνυται ή παθαίνει ζημιά.

Για αυτό τον λόγο υπάρχουν κάποιες προσθήκες σε σχέση με την GameObject, όπως:

ObjImageArray

Η ObjImageArray έρχεται αντιστοίχως στην θέση της ObjImage της GameObject. Η ObjImageArray δέχεται έναν πίνακα αντικειμένων bitmap , στον οποίο τα στοιχεία 0 έως 3 αναπαριστούν τα frames της κανονικής κατάστασης του αντικειμένου, τα frames 4 έως 7 την κατάσταση έκρηξης του αντικειμένου και τα frames 8 έως 11 την κατάσταση ζημιάς του αντικειμένου.

Frames για τις καταστάσεις animation «Normal» και «Explosion» του εχθρικού UFO.

ObjAnimation

Η ObjAnimation είναι τύπου Animation Status και χρησιμοποιείται για να δείχνει σε τι κατάσταση βρίσκεται το αντικείμενο.

Το enum AnimationStatus έχει τις τιμές Normal, Explosion, Damage ως εξής:

```
Enum AnimationStatus As Integer
```

```
        Normal = 0
        Explosion = 1
        Damage = 2

End Enum
```

AnimationCounterNormal, AnimationCounterExplosion, AnimationCounterDamaged

Αυτές οι μεταβλητές χρησιμοποιούνται ως μετρητές για την εμφάνιση του επόμενου frame σε κάθε κατάσταση στον σωστό χρόνο.

2.2.4 Λειτουργίες της κλάσης GameObjectAnimated

DrawObject

Η DrawObject της GameObjectAnimated δουλεύει όπως και η αντίστοιχη της GameObject με την διαφορά ότι προηγείται ένα κομμάτι κώδικα το οποίο κάθε φορά αναλαμβάνει να δώσει την σωστή τιμή στην μεταβλητή DisplayFrameNumber η οποία δείχνει ποιο frame από τον πίνακα ObjImageArray πρέπει να χρησιμοποιηθεί. Ανάλογα με την κατάσταση του ObjAnimation του αντικειμένου εκτελείται και το αντίστοιχο κομμάτι κώδικα.

Πιο αναλυτικά:

Κανονική κατάσταση

```
If ObjAnimation = AnimationStatus.Normal Then

                DisplayFrameNumber = AnimationCounterNormal / GameFPS

                AnimationCounterNormal += 4

                If AnimationCounterNormal > (GameFPS * 3) Then
AnimationCounterNormal = 0

End If
```

Εδώ το DisplayFrameNumber θα πάρει τιμές από 0 έως 3. Στον αντίστοιχο μετρητή προστίθενται +4 κάθε φορά. Αφού όπως έχει ήδη εξηγηθεί η σταθερά GameFPS καθορίζει πόσες φορές το δευτερόλεπτο ο κώδικας θα φτάνει σε αυτό το σημείο, βλέπουμε για παράδειγμα:

Εάν έχουμε GameFPS = 24 τότε ο κώδικας θα εκτελεστεί 24 φορές το δευτερόλεπτο. Αυξάνοντας το μετρητή κατά 4 κάθε φορά αυτό σημαίνει ότι κάνοντας DisplayFrameNumber = AnimationCounterNormal / GameFPS η DisplayFrameNumber θα αυξάνεται κατά 1 κάθε 4 frames, δηλαδή κάθε 0,16 sec. (1 sec / 24) * 4 = 0,16 sec

Αν αυξάναμε όχι κατά +4 αλλά κατά +8 τον μετρητή AnimationCounterNormal τότε το animation θα ήταν πιο αργό, με διαφορετικό frame κάθε 0,33 sec. Πιο αργό animation

χρησιμοποιείται στις καταστάσεις Explosion και Damage για καλύτερο “ορατό” αποτέλεσμα. Η if AnimationCounterNormal > (GameFPS * 3) μηδενίζει τον μετρητή καθώς τα frames της κάθε κατάστασης είναι 4 (0,1,2,3 = Normal)

Κατάσταση έκρηξης (θανάτου του αντικειμένου)

```
If ObjAnimation = AnimationStatus.Explosion Then

                DisplayFrameNumber = (AnimationCounterExplosion /
GameFPS) + 4

                AnimationCounterExplosion += 12

                If AnimationCounterExplosion > (GameFPS * 3) Then

                    AnimationCounterExplosion = 0

                    ObjAnimation = AnimationStatus.Normal

                    IsAlive = False

                End If

End If
```

Παρομοίως και στην κατάσταση Explosion πετυχαίνουμε την ακολουθία τιμών από 4 έως 7 για την DisplayFrameNumber, με βασική λειτουργία την “απενεργοποίηση” (IsAlive=False) του αντικειμένου όταν το animation της έκρηξης τελειώσει.

Κατάσταση ζημιάς (τραυματισμού του αντικειμένου)

```
If ObjAnimation = AnimationStatus.Damage Then

                DisplayFrameNumber = (AnimationCounterDamaged /
GameFPS) + 8

                AnimationCounterDamaged += 12

                If AnimationCounterDamaged > (GameFPS * 3) Then

                    AnimationCounterDamaged = 0

                    ObjAnimation = AnimationStatus.Normal

                End If

End If
```

Στην κατάσταση Damage επίσης ορίζεται η κατάσταση του αντικειμένου πάλι σε Normal όταν τελειώσει το Damage animation. Οι τιμές του DisplayFrameNumber που προκύπτουν είναι από 8 έως 11.

Στο τέλος η DrawImage δουλεύει όπως και στην GameObject, χρησιμοποιώντας το σωστό frame από τον πίνακα ObjImageArray(DisplayFrameNumber)

```
Dim g As Graphics = Graphics.FromImage(bmpDoubleBuffer)

      dstRect = New Rectangle(ObjX, ObjY, ObjImageArray(0).Width,
ObjImageArray(0).Height)
      TransparencyAttr.SetColorKey(Color.Red, Color.Red)
      g.DrawImage(ObjImageArray(DisplayFrameNumber), dstRect, 0, 0,
ObjImageArray(0).Width, ObjImageArray(0).Height, GraphicsUnit.Pixel,
TransparencyAttr)
```

2.3 Βοηθητικές λειτουργίες

Κώδικες οι οποίοι ήταν αναγκαίοι σε διάφορα σημεία του project υλοποιούνται με βοηθητικές συναρτήσεις.

ShowGFXCentered

Παίρνει ως όρισμα ένα αντικείμενο τύπου bitmap και το γράφει στο bmpDoubleBuffer κεντραρισμένο. Χρησιμοποιείται κυρίως στα Splash screens στην αρχή κάθε πίστας. Η κλήση της συνάρτησης πρέπει να ακολουθείται και από ένα Invalidate της φόρμας έτσι ώστε το αποτέλεσμα να είναι ορατό.

ShowGFXAtXY

Παρομοίως με την ShowGFXCentered μόνο που εδώ παίρνει και ως όρισμα συγκεκριμένες συντεταγμένες όπου θα εμφανιστεί το bitmap. Χρησιμοποιείται για την σχεδίαση του status bar του παίκτη που βρίσκεται στο κάτω μέρος της οθόνης με την ενέργεια και την πρόοδο της πίστας.

FreezeGameForXSeconds

Παγώνει το παιχνίδι για τον αριθμό δευτερολέπτων που θα δοθούν, απενεργοποιώντας και ενεργοποιώντας ξανά τον Timer1 που είναι υπεύθυνος για την ροή του παιχνιδιού. Χρησιμοποιεί το αντικείμενο timer2 για να μετράει την διάρκεια της παύσης.

BitmapStreamLoader

Φορτώνει από τα resources του project τις bmp εικόνες σε bitmap αντικείμενα. Προσοχή: Χρησιμοποιεί την σταθερά ProjectName η οποία πρέπει να έχει ως τιμή το όνομα του project.

Πιο αναλυτικά:

```
Public Function BitmapStreamLoader(ByVal StreamName As String) As
Bitmap

         Dim TempStream As System.IO.Stream =
Me.GetType().Assembly.GetManifestResourceStream((Trim(ProjectName) +
"." + StreamName))

        Return New Bitmap(TempStream)

End Function
```

Το όρισμα που δέχεται είναι ένα string, το οποίο είναι το όνομα του bmp που βρίσκεται στα resources. Χρησιμοποιώντας το αντικείμενο TempStream τύπου Stream φορτώνουμε το bmp σε αυτό και το χρησιμοποιούμε για να κατασκευάσουμε ένα νέο bitmap το οποία θα περιέχει το bmp.

Προσοχή στα bitmaps που εισάγονται στα resources, καθώς το Build Action τους θα πρέπει να είναι Embedded Resource.

ObjRandomizer

Παίρνει ως ορίσματα δύο ακέραιες τιμές, και επιστρέφει μια τυχαία τιμή μέσα στο εύρος αυτών των τιμών.

Αξίζει να σημειωθεί ότι η μεταβλητή r τύπου random δημιουργείται έξω από την συνάρτηση για να έχουμε καλύτερα αποτελέσματα, κάνοντας μια φορά “εκκίνηση” στην γεννήτρια ψευδό-τυχαίων αριθμών. Για αυτό το λόγο η εντολή

```
Dim r As Random = New Random()
```

Βρίσκεται στην αρχή του κώδικα της Form1 και όχι μέσα στην ObjRandomizer

2.4 Περιγραφή μεταβλητών και σταθερών

2.4.1 Γενικές μεταβλητές και σταθερές

Διάφορες σταθερές του παιχνιδιού

CUSTOM_LEVEL_MODE

Αν είναι true εμφανίζεται το Panel δημιουργίας custom πίστας, αν είναι false το παιχνίδι τρέχει αμέσως με τις προκατασκευασμένες πίστες.

MAXIMUMENEMYPERTYPE

Καθορίζει το μέγιστο δυνατό αριθμό ταυτόχρονων αντιπάλων ανά τύπο που μπορεί να δηλωθεί σε μια πίστα. Ουσιαστικά χρησιμοποιείται για τη δέσμευση μνήμης για τους πίνακες αντικειμένων τύπου EnemyObject.

MAXIMUMLASERPERTYPE

Καθορίζει το μέγιστο δυνατό αριθμό ταυτόχρονων laser ανά αντίπαλο που μπορεί να δηλωθεί σε μια πίστα. Ουσιαστικά χρησιμοποιείται για τη δέσμευση μνήμης για τους πίνακες αντικειμένων τύπου EnemyMissile

NumberOfLevelsCONST

Σταθερά που δείχνει τον αριθμό τον ήδη προγραμματισμένων πιστών.

ProjectName

Το όνομα του project. Προσοχή στην τιμή της η οποία πρέπει να είναι σωστή. Χρησιμοποιείται από την bitmapStreamLoader.

GameFPS

Καθορίζει πόσες φορές το δευτερόλεπτο θα “ανανεώνεται” το παιχνίδι. Ο ενδεικτικός αριθμός των 24 frames per second είναι κοινώς αποδεκτός για παιχνίδια τέτοιου τύπου καθώς έχουμε ωραίο αποτέλεσμα χωρίς κατάχρηση πόρων. (Μοντέρνα παιχνίδια σε desktop συστήματα φτάνουν και > 60 fps). Μπορούμε να αυξήσουμε περισσότερο την τιμή αλλά ανάλογα με το σύστημα ίσως το παιχνίδι να “κολλάει” ενώ ταυτόχρονα το καλύτερο αποτέλεσμα δεν είναι ιδιαίτερα εμφανές στο ανθρώπινο μάτι. Οπότε 24 είναι μια αρκετά καλή επιλογή.

PlayerMaxHealth

Η μέγιστη υγεία του παίκτη.

PlayerLasernum

Τα laser που μπορεί να έχει πυροβολήσει ο παίκτης ταυτόχρονα στην οθόνη.

EnergyBonus

Πόσο κέρδος στην υγεία του παίκτη έχει το bonus.

EnergyPossibility

Πόσο πιθανό όταν σκοτώνεται ένας αντίπαλος να εμφανιστεί το bonus. Μικρή τιμή, μικρή πιθανότητα. Μια καλή τιμή είναι 5.

StarsNum

Αριθμός των μεγάλων αστεριών του background

MinStarSpeed

Ελάχιστη ταχύτητα των μεγάλων αστεριών του background

MaxStarSpeed

Μέγιστη ταχύτητα των μεγάλων αστεριών του background

SmallStarsNum

Αριθμός των μικρών αστεριών του background

MinSmallStarSpeed

Ελάχιστη ταχύτητα των μικρών αστεριών του background

MaxSmallStarSpeed

Μέγιστη ταχύτητα των μικρών αστεριών του background

GameCompleted

Boolean που δείχνει εάν ο παίκτης έχει ολοκληρώσει το παιχνίδι νικώντας στην τελευταία πίστα

GameOver

Boolean που δείχνει εάν ο παίκτης έχει χάσει.

bmpDoubleBuffer

Το αντικείμενο τύπου bitmap πάνω στο οποίο γίνεται η σχεδίαση όλων των αντικειμένων του παιχνιδιού.

r

Τύπου random, χρησιμοποιείται από την ObjRandomizer.

Διάφορα γραφικά

Τα παρακάτω γραφικά φορτώνονται με την βοήθεια της λειτουργίας bitmapStreamLoader:

CurrentBackground

Αντικείμενο τύπου bitmap το οποίο έχει το τρέχον background.

GameOverBMP

Η εικόνα που φαίνεται όταν ο παίκτης χάσει.

GameCompletedBMP

Η εικόνα που φαίνεται όταν ο παίκτης νικήσει το παιχνίδι.

MissionAccomplishedBMP

Η εικόνα που φαίνεται όταν ο παίκτης νικήσει μια πίστα.

StarBMP

Η εικόνα των μεγάλων αστεριών του background.

SmallStarBMP

Η εικόνα των μικρών αστεριών του background.

PlayerLaserBMP

Η εικόνα του laser του παίκτη.

MissionTextBMP, EnergyTextBMP, GreenBar, BlueBar, GreyBar, RedBar

Γραφικά της μπάρας κατάστασης του παίκτη.

Μεταβλητές των διάφορων αντικειμένων και των πιστών

Player

Το διαστημόπλοιο του παίκτη.

PlayerLaserShots(PlayerLaserNum)

Ένας πίνακας αντικειμένων τύπου PlayerMissile. Τα laser του παίκτη.

Energy

Το bonus ενέργειας.

BackStars(StarsNum), BackSmallStars(SmallStarsNum)

Πίνακες αντικειμένων τύπου StarObject, είναι τα μεγάλα και μικρά αστέρια του background.

GameLevels(NumberOfLevels)

Πίνακας αντικειμένων τύπου GameLevel. Είναι οι διάφορες πίστες του παιχνιδιού. Οι θέσεις που δεσμεύονται ορίζονται από την σταθερά NumberOfLevels.

Έλεγχος πληκτρολόγιου:

KeyFlag

Η KeyFlag χρησιμοποιείται στην ρουτίνα που ελέγχει τα κουμπιά που πατιούνται στο πληκτρολόγιο και στην κίνηση του διαστημόπλοιου του παίκτη. Οι τιμές που χρησιμοποιούνται είναι του enum Directions.

Enum Directions

Το enum Directions έχει τις εξής διαθέσιμες τιμές:

NullDirection = 0
GoUp = 1
GoDown = 2
GoLeft = 3
GoRight = 4
Fire = 5
NullDirection_FireException = 6
RESERVED_FUTURE_USE = 7
GoUpStop = 8
GoDownStop = 9
GoLeftStop = 10
GoRightStop = 11
FireStop = 12

Οι τιμές 1 εώς 5 χρησιμοποιούνται όταν πατιέται κάποιο πλήκτρο (KeyDown), οι 8 εώς 12 όταν αφήνεται ένα πλήκτρο (KeyUp).

2.4.2 Μεταβλητές προγραμματισμού και ελέγχου των πιστών του παιχνιδιού

CustomUFOProperties, CustomAlienProperties, CustomMeteoriteProperties

Αυτές οι μεταβλητές είναι τύπου EnemyProperties και οι διάφορες τιμές τους ορίζονται από την οθόνη κατασκευής custom πίστας με σκοπό να χρησιμοποιηθούν κατά την κλήση του Constructor ενός αντικειμένου GameLevel.

CustomKillsCount

Ο αριθμός των εχθρών που πρέπει να καταστρέψει ο παίκτης για να νικήσει την custom made πίστα.

CurrentLevel

Δείχνει τον αριθμό της τρέχουσας πίστας.

NumberOfLevels

Ο αριθμός των πιστών. Αρχικά παίρνει την τιμή του NumberOfLevelsCONST. Εάν όμως ο χρήστης επιλέξει να παίξει μια custom πίστα, αυτή η μεταβλητή παίρνει την τιμή 1.

LevelProgress

Αυτή η μεταβλητή μετράει την πρόοδο της πίστας, δηλαδή το πόσο έχει προχωρήσει ο παίκτης μέχρι τον στόχο της πίστας.

LevelProgressSTEP

Δείχνει πόσα τετραγωνάκια στην μπάρα προόδου της πίστας αντιστοιχούν για μια καταστροφή εχθρού.

Enum AnimationStatus

Χρησιμοποιείται για τον ορισμό της κατάστασης του animation ενός GameObjectAnimated. Οι διαθέσιμες τιμές είναι:

Normal = 0
Explosion = 1
Damage = 2

Η δομή EnemyProperties

Χρησιμοποιώντας την δομή EnemyProperties ο προγραμματιστής μπορεί εύκολα να προγραμματίσει τα χαρακτηριστικά που θέλει να έχει κάποιος εχθρός και να κατασκευάσει με αυτόν τον τρόπο διαφορετικές πίστες χρησιμοποιώντας μεταβλητές τύπου EnemyProperties ως ορίσματα στον Constructor ενός αντικειμένου τύπου GameLevel.

Η δομή EnemyProperties είναι η εξής:

```
Public Structure EnemyProperties

    Dim Enabled As Boolean
    Dim EnemyAnimation() As Bitmap
    Dim EnemyX, EnemyY As Integer
    Dim EnemyMinX, EnemyMinY As Integer
    Dim EnemyMaxX, EnemyMaxY As Integer
    Dim EnemyMinSpeed, EnemyMaxSpeed As Integer
    Dim EnemyAgressiveness As Integer
    Dim EnemyCount As Integer
    Dim EnemyFireNum As Integer
    Dim EnemyLaserDamage As Integer
    Dim EnemyCollisionDamage As Integer
```

```
End Structure
```

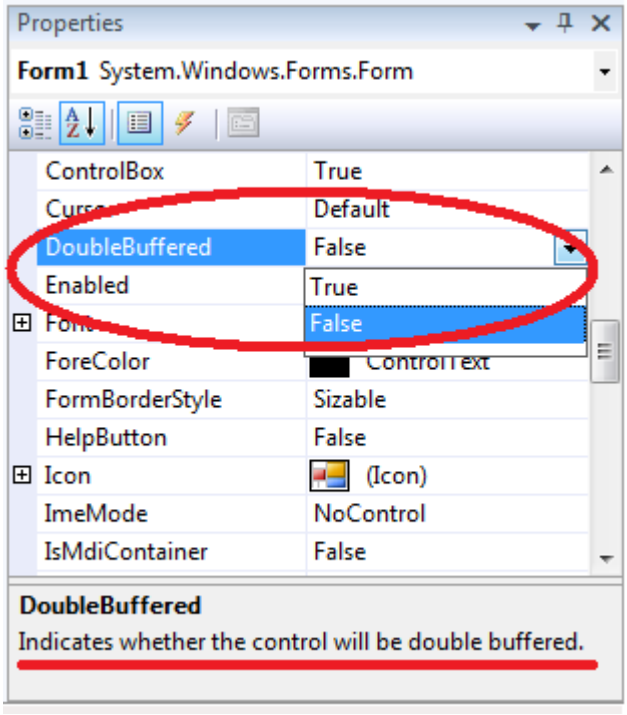
Το Boolean Enabled δείχνει εάν ο εχθρός υπάρχει ή όχι. Το EnemyAnimation είναι ο πίνακας αντικειμένων τύπου bitmap που είναι τα frames του animation του εχθρού και θα πρέπει να υπόκεινται στους κανόνες 0-3 Normal, 4-7 Explosion, 8-11 Damage. Το EnemyX, EnemyY είναι η αρχική θέση του εχθρού, ενώ τα EnemyMinX, EnemyMinY, EnemyMaxX, EnemyMaxY καθορίζουν την περιοχή κίνησης του. Τα EnemyMinSpeed και EnemyMaxSpeed καθορίζουν την ελάχιστη και την μέγιστη ταχύτητα που μπορεί να έχει ο εχθρός. Το EnemyAgressiveness ορίζει την επιθετικότητα του εχθρού. Μια υψηλή τιμή θα κάνει τον εχθρό να πυροβολεί συχνά, ενώ το πόσα laser θα μπορεί να έχει ανά πάσα στιγμή ταυτόχρονα στην οθόνη εξαρτάται από το EnemyFireNum. Το EnemyCount ορίζει πόσοι τέτοιοι εχθροί θα υπάρχουν ταυτόχρονα στην πίστα. Η ζημιά που θα δέχεται ο παίκτης σε περίπτωση που θα χτυπηθεί από το laser του εχθρού ορίζεται από την EnemyLaserDamage ενώ η ζημιά σε περίπτωση σύγκρουσης με τον εχθρό από την EnemyCollisionDamage.

2.5 Σχεδίαση γραφικών / Flickering

Ένα πρόβλημα το οποίο δημιουργήθηκε κατά την ανάπτυξη του παιχνιδιού ήταν το λεγόμενο flickering.

Flickering σε μια τυπική οθόνη CRT

Το flickering αντιμετωπίζεται ανάλογα με την περίπτωση είτε με αύξηση του refresh rate της οθόνης είτε με μεθόδους double buffering. Σε εφαρμογές που αναπτύσσονται στο «κανονικό» .NET Framework, οι φόρμες έχουν την δυνατότητα Double Buffering. Το πρόβλημα ήταν η απουσία του double buffering από το .NET Compact Framework.



Ενσωματωμένη λειτουργία Double Buffering σε .NET Framework, όχι όμως και στο .NET Compact Framework

Δυστυχώς, το .NET Compact Framework μέχρι την τρέχουσα έκδοση 3.5 δεν προσφέρει ενσωματωμένη αυτή την λειτουργία. Η μοναδική λύση είναι να υλοποιηθεί μια double buffer λειτουργία.

Αρχικά, κάνουμε Override την OnPaintBackground της φόρμας μας και κάνουμε comment-out την MyBase.OnPaintBackground(e) .

```
Protected Overrides Sub OnPaintBackground(ByVal e As
System.Windows.Forms.PaintEventArgs)

        ' MyBase.OnPaintBackground(e)

End Sub
```

Στην συνέχεια θα κάνουμε Override την OnPaint ενώ θα κατασκευάσουμε και μια λειτουργία που θα ονομάσουμε CreateDoubleBuffer:

```
Protected Overrides Sub OnPaint(ByVal e As
System.Windows.Forms.PaintEventArgs)

        CreateDoubleBuffer()  ' Create double buffering bitmap

        e.Graphics.DrawImage(bmpDoubleBuffer, 0, 0)

End Sub
```

Η CreateDoubleBuffer είναι η εξής:

```
Private Sub CreateDoubleBuffer()


        If bmpDoubleBuffer Is Nothing Then

            bmpDoubleBuffer = New Bitmap(Me.Width, Me.Height) ' Full
Screen buffering
            Dim g As Graphics = Graphics.FromImage(bmpDoubleBuffer) '
kentriko buffer
            g.DrawImage(CurrentBackground, 0, 0)
        End If


End Sub
```

Το Graphics πάνω στο οποίο θα γίνει το Double Buffering είναι το g το οποίο παίρνει σαν όρισμα το bitmap bmpDoubleBuffer και δημιουργείται αν δεν υπάρχει (If bmpDoubleBuffer is Nothing).

Με λίγα λόγια, όλοι οι σχεδιασμοί των αντικειμένων θα γίνονται στην μνήμη, πάνω στο αντικείμενο bmpDoubleBuffer και όχι απευθείας στην οθόνη και μόνο όταν όλα τα αντικείμενα έχουν τελειώσει με τον σχεδιασμό των νέων θέσεών τους, θα καλείται μια ανανέωση της οθόνης με το bmpDoubleBuffer (το οποίο και καταλαμβάνει όλη την οθόνη). Δηλαδή το αντικείμενο που θα σχεδιάζεται στην οθόνη θα είναι το ένα bmpDoubleBuffer (μέσα στο οποίο έχουν ήδη σχεδιαστεί στην μνήμη όλα τα επιμέρους αντικείμενα). Αυτό λύνει το πρόβλημα του flickering.

2.6 Διάβασμα από το πληκτρολόγιο

Για την ανάγνωση του κουμπιού που πατήθηκε από το πληκτρολόγιο χρησιμοποιούμε τις KeyUp/KeyDown σε συνδυασμό με την shared μεταβλητή KeyFlag.

Ανάλογα με το πιο πλήκτρο πατιέται, δίνουμε στην KeyFlag την σωστή Directions τιμή, ενώ το ίδιο συμβαίνει και όταν ένα πλήκτρο αφήνεται. Ένα μικρό ενδεικτικό κομμάτι κώδικα είναι το εξής από την KeyDown και αφορά το πλήκτρο «Πάνω βέλος»:

```
If (e.KeyCode = System.Windows.Forms.Keys.Up) Then

            KeyFlag = Directions.GoUp

End If
```

Το KeyFlag παρακάτω μέσα στον timer1 περνάει στην PositionUpdate του αντικειμένου του παίκτη, η οποία παίρνει τον έλεγχο από εκεί και πέρα και αναλαμβάνει να κινήσει το διαστημόπλοιο του παίκτη.

2.7 Περιγραφή λειτουργιών και κλάσεων

2.7.1 Form1 Load

Στην Load της form1 ελέγχεται εάν το παιχνίδι είναι σε CUSTOM_LEVEL_MODE ή όχι. Εάν είναι τότε εμφανίζεται το πάνελ παραμετροποίησης της custom πίστας, διαφορετικά το παιχνίδι ξεκινάει με τις προκαθορισμένες πίστες περνώντας τον έλεγχο στην Init.

2.7.2 Form1 Init

Στην Init συμβαίνουν όλες οι απαραίτητες αρχικοποιήσεις του παιχνιδιού. Φορτώνονται όλα τα γραφικά του παιχνιδιού χρησιμοποιώντας την βοήθεια της bitmapStreamLoader, ενώ κατασκευάζονται οι απαραίτητες μεταβλητές τύπου EnemyProperties με τις οποίες θα προγραμματιστούν οι πίστες του παιχνιδιού.

Ενδεικτικά οι τιμές για τον εχθρό τύπου “UFO”

```
DefaultUFOProperties.Enabled = True
DefaultUFOProperties.EnemyAgressiveness = 15
DefaultUFOProperties.EnemyAnimation = UFOAnimated
DefaultUFOProperties.EnemyCollisionDamage = 10
DefaultUFOProperties.EnemyCount = 3
DefaultUFOProperties.EnemyFireNum = 2
DefaultUFOProperties.EnemyLaserDamage = 5
DefaultUFOProperties.EnemyMaxSpeed = 3
DefaultUFOProperties.EnemyMaxX = Me.Width
DefaultUFOProperties.EnemyMaxY = Me.Height / 2 +
UFOAnimated(0).Height
DefaultUFOProperties.EnemyMinSpeed = 2
DefaultUFOProperties.EnemyMinX = 0
DefaultUFOProperties.EnemyMinY = 0
DefaultUFOProperties.EnemyX = 0
DefaultUFOProperties.EnemyY = 0
```

Στις συγκεκριμένες ρυθμίσεις βλέπουμε ότι τα UFO υπάρχουν (Enabled) έχουν επιθετικότητα 15, τα γραφικά τους είναι μέσα στο array UFOAnimated, σε περίπτωση σύγκρουσης ο παίκτης χάνει 10 από την ενέργειά του ενώ σε περίπτωση που χτυπηθεί με το laser τους χάνει 5, τα UFO μπορούν να ρίξουν μέχρι 2 laser ανά πάσα στιγμή, τα UFO θα είναι 3 σε αριθμό, η περιοχή κίνησής τους είναι όλο το πλάτος της οθόνης και λίγο περισσότερο από το μισό ύψος της οθόνης, η ελάχιστη ταχύτητά τους είναι 2 pixel ανά ανανέωση ενώ η μέγιστη 3 και η αρχική τους θέση 0,0.

Στη συνέχεια στην Init ορίζεται το Interval του timer1 σύμφωνα με την σταθερά GameFPS ο οποίος ουσιαστικά τρέχει το παιχνίδι.

Ακολουθούν οι κλήσεις των constructors των αντικειμένων του παιχνιδιού τα οποία αποτελούν στοιχεία σε αντίστοιχους πίνακες αντικειμένων.

Για παράδειγμα η δημιουργία του διαστημόπλοιου του παίκτη:

```
' Dimiourgia paikti
Player = New PlayerObject(PlayerAnimated, ((Me.Width / 2) -
(PlayerAnimated(0).Width / 2)), ((Me.Height) -
(PlayerAnimated(0).Height)), 0, 0, Me.Width, Me.Height - 20, 4, 4)
```

Τα ορίσματα αφορούν τα γραφικά του παίκτη, την αρχική του θέση, την περιοχή κίνησής του και την ταχύτητά του.

Πολύ σημαντικό είναι το σημείο όπου κατασκευάζονται οι πίστες. Εδώ ουσιαστικά φαίνεται το πόσο εύκολα μπορεί να δημιουργηθεί/παραμετροποιηθεί μια πίστα, αποτέλεσμα της προγραμματιστικής προσέγγισης του παιχνιδιού. Το απόσπασμα του κώδικα που κατασκευάζει τις 3 πίστες του παιχνιδιού είναι το εξής:

```
' Dimiourgia ton prokathorismenon piston paixnidiou
GameLevels(0) = New GameLevel(LevelSplashes(0),
BitmapStreamLoader("Galaxy1.bmp"), 5, DefaultUFOProperties,
NullProperties, DefaultMeteoriteProperties)

GameLevels(1) = New GameLevel(LevelSplashes(1),
BitmapStreamLoader("Galaxy3.bmp"), 10, NullProperties,
DefaultAlienProperties, DefaultMeteoriteProperties)

GameLevels(2) = New GameLevel(LevelSplashes(2),
BitmapStreamLoader("Galaxy4.bmp"), 25, DefaultUFOProperties,
DefaultAlienProperties, DefaultMeteoriteProperties)
```

Όπως βλέπουμε, μια πίστα σαν ορίσματα χρειάζεται μόνο το γραφικό που θα έχει ως background, το πόσους εχθρούς πρέπει να καταρίψει ο παίκτης για να τελειώσει η πίστα και τα χαρακτηριστικά των 3 τύπων εχθρών UFO, Alien, Meteorites. Μπορούμε να περάσουμε τα NullProperties εάν θέλουμε να απενεργοποιήσουμε έναν εχθρό. Στις συγκεκριμένες πίστες χρησιμοποιήσαμε για κάθε εχθρό τα ίδια EnemyProperties και απλώς παίξαμε με την παρουσία η όχι των εχθρών σε κάθε πίστα για να αυξήσουμε δυσκολία. Θα μπορούσαμε όμως να προγραμματίσουμε διαφορετικά EnemyProperties π.χ. για τα UFO της 1ης πίστας, διαφορετικά (πιο δύσκολα) πάλι για τα UFO αλλά της 2ης πίστας.

2.7.3 Timer1

Το timer1 είναι υπεύθυνο για την ροή του παιχνιδιού. Αποτελείται από ένα μεγάλο block if then else (με ακόμα περισσότερα εμφωλευμένα) το οποίο καλεί ανάλογα με την περίπτωση τις σωστές λειτουργίες. Χωρίζεται σε δύο μπλόκ, αυτό του then και αυτό του else σύμφωνα με την συνθήκη

```
If Not GameOver And Not GameCompleted
```

Η boolean GameOver παίρνει την τιμή true όταν το διαστημόπλοιο του παίκτη καταστραφεί (χάσει όλη του την ενέργεια) ενώ η boolean GameCompleted παίρνει την τιμή true όταν ο παίκτης τελειώσει επιτυχώς την τελευταία πίστα του παιχνιδιού.

Ας δούμε αναλυτικά και τα δύο αυτά μπλόκ:

Το block “then”

Σε περίπτωση που δεν έχουμε GameOver και δεν έχουμε GameCompleted του παιχνιδιού, το παιχνίδι εκτελείται «φυσιολογικά».

```
' Leitourgia tis trexousas pistas (Kinisi, sxediasi exthron k.l.p.)
If CurrentLevel - 1 = NumberOfLevels Then GameCompleted = True Else
GameLevels(CurrentLevel - 1).Update()
```

Ξεκινάμε με την κλήση της σωστής πίστας GameLevel του παιχνιδιού από το αντίστοιχο array που τις περιέχει. Η μεταβλητή CurrentLevel έχει την τιμή 1 όταν είμαστε στην πρώτη πίστα, 2 στη δεύτερη κ.λ.π. Οπότε χρησιμοποιούμε την CurrentLevel – 1 για να δείξουμε στο σωστό στοιχείο του array καθώς η αρίθμηση των στοιχείων του array ξεκινάει από το 0 και όχι από το 1. Η λειτουργία Update των αντικειμένων τύπου GameLevel είναι υπεύθυνη για τον έλεγχο και την σχεδίαση όλων των εχθρών της πίστας.

Το CurrentLevel αυξάνεται κάθε φορά που τελειώνει επιτυχώς μια πίστα, που σημαίνει ότι στην τελευταία πίστα το CurrentLevel θα έχει την τιμή NumberOfLevels. Αυτό σημαίνει ότι ο παίκτης κέρδισε την τελευταία πίστα, οπότε σηκώνουμε την σημαία GameCompleted ως true.

Στην συνέχεια έχουμε εφόσον το παιχνίδι δεν είναι GameCompleted τις ανανεώσεις των θέσεων και την σχεδίαση των αστεριών του background, του παίκτη, της μπάρας κατάστασης και των laser του παίκτη:

```
' Metakiniseis antikeimenon kai sxediasi auton

If Not GameCompleted Then

                ' Asteria megala tou background...
                For i As Integer = 0 To StarsNum - 1
                    BackStars(i).PositionUpdate()
                    BackStars(i).DrawObject()
                Next i

                ' ...kai ta mikra
                For i As Integer = 0 To SmallStarsNum - 1
                    BackSmallStars(i).PositionUpdate()
                    BackSmallStars(i).DrawObject()
                Next i
```

```
                ' Kinisi / Sxediasi paikti kai tou status tou
                Player.PositionUpdate()
                Player.DrawObject()
                Player.PlayerStatus()


                ' Kinisi / Sxediasi ton laser tou paikti
                For i As Integer = 0 To PlayerLaserNum - 1
                    PlayerLaserShots(i).PositionUpdate()
                    If PlayerLaserShots(i).IsAlive Then

                        PlayerLaserShots(i).DrawObject()
                    End If
                Next i


End If

            ' Me auton ton tropo prokaloume tin ektelesi tis OnPaint
gia tin sxediasi ton grafikon
            Me.Invalidate()
```

Η Me.Invalidate προκαλεί την εκτέλεση της OnPaint της form1 στην οποία έχουμε αναφερθεί προηγουμένως στο double buffering.

Το block "else"

Εδώ οδηγούμαστε όταν έχουμε κατάσταση GameOver ή GameCompleted.

```
Else

            'Edo tha vrethoume otan o paiktis tha exei xasei i tha
exei teleiosei to paixnidi
            'Deixnoume to analogo minima...
            If GameOver Then ShowGFXCentered(GameOverBMP) Else
ShowGFXCentered(GameCompletedBMP)

            ' Ektelesi OnPaint...
            Me.Invalidate()

            ' Pagoma paixnidiou gia 3 deuterolepta
            FreezeGameForXSeconds(3)

            ' Epanekkinisi paixnidiou stin proti pista. Arxikopoiisi
kapoion metavliton elegxou k.a.
            GameOver = False
            GameCompleted = False
            Player.IsAlive = True
            Player.ObjAnimation = AnimationStatus.Normal
            Player.Health = Player.MaxHealth
            For i As Integer = 0 To NumberOfLevels - 1
                GameLevels(i).Init()
                GameLevels(i).InitedOnce = False
            Next
            LevelProgress = 0
            CurrentLevel = 1
```

```
End If
```

Δείχνουμε το ανάλογο splash screen (Game over ή αυτό του τερματισμού του παιχνιδιού) και μετά αρχικοποιούμε όλες τις απαραίτητες μεταβλητές έτσι ώστε το παιχνίδι να ξεκινήσει και πάλι από την πρώτη πίστα κανονικά.

2.7.4 Η κλάση GameLevel

Η GameLevel είναι μια κλάση η οποία υλοποιεί μια πίστα του παιχνιδιού. Ενσωματώνει όλες εκείνες τις λειτουργίες που δημιουργούν τις δυνατότητες εύκολης παραμετροποίησης και δημιουργίας διαφορετικών πιστών του παιχνιδιού από τον προγραμματιστή. Περιέχει πλήθος μεταβλητών και λειτουργιών καθώς ασχολείται και με τον έλεγχο κίνησης και την σχεδίαση των εχθρών, όπως αντίστοιχα καλούνται οι ανάλογες λειτουργίες του παίκτη από την timer1.

Μεταβλητές

```
Public Class GameLevel

        Public UFOProperties, AlienProperties, MeteoriteProperties As
EnemyProperties
        Public UFOArray(MAXIMUMENEMYPERTYPE),
AliensArray(MAXIMUMENEMYPERTYPE) As EnemyObject
        Public MeteoritesArray(MAXIMUMENEMYPERTYPE) As
MeteoriteObject
        Public IntroSplash As Bitmap
        Public IntroSplashShown As Boolean
        Public MissionKills, KillsCount As Integer
        Public InitedOnce As Boolean = False
        Public LevelBackground As Bitmap
        Public IncreasedLevelOnce As Boolean = False

        Public MissionAccomplished As Boolean = False
```

Η GameLevel ξεκινάει με την δήλωση κάποιων μεταβλητών. Οι πρώτες 3 μεταβλητές τύπου EnemyProperties παίρνουν τιμές από τα ορίσματα που θα δοθούν στον constructor και ουσιαστικά είναι οι παράμετροι των εχθρών του παιχνιδιού. Στη συνέχεια δεσμεύονται θέσεις μνήμης για τους απαραίτητους πίνακες των αντικειμένων (UFOArray, AliensArray, MeteoritesArray. Το IntroSplash είναι το εισαγωγικό γραφικό που φαίνεται στην αρχή της πίστας, ενώ το IntroSplashShown δείχνει εάν αυτό έχει εμφανιστεί η όχι. Τα MissionKills, KillsCount αντιστοίχως αναφέρονται στο πόσους εχθρούς πρέπει να καταρρίψει ο παίκτης για να κερδίσει την πίστα και πόσους έχει καταρρίψει μέχρι στιγμής. Το LevelBackground είναι ένα αντικείμενο τύπου bitmap το οποίο είναι το background της πίστας. Το Boolean IncreasedLevelOnce δείχνει εάν έχει αυξηθεί ήδη κατά 1 ο αριθμός της τρέχουσας πίστας (όταν θα ο παίκτης τελειώσει αυτή την πίστα ο μετρητής της τρέχουσας πίστας θα πρέπει να αυξηθεί μια φορά κατά 1 έτσι ώστε να δείχνει στην επόμενη). Το Boolean MissionAccomplished δείχνει εάν ο παίκτης έχει ολοκληρώσει επιτυχώς την πίστα.

Constructor

Ο constructor παίρνει ως ορίσματα τα εξής:

```
Public Sub New(ByVal A_IntroSplash As Bitmap, ByVal A_LevelBackground
As Bitmap, ByVal A_MissionKills As Integer, ByVal A_UFOProperties As
EnemyProperties, ByVal A_AlienProperties As EnemyProperties, ByVal
A_MeteoriteProperties As EnemyProperties)
```

A_IntroSplash: Ένα αντικείμενο τύπου bitmap το οποίο είναι το splash screen που φαίνεται για λίγο στην αρχή της πίστας. Για παράδειγμα ένα bitmap που γράφει «Πίστα 1: Γαλαξίας Ανδρομέδας»

A_LevelBackground: Ένα αντικείμενο τύπου bitmap το οποίο είναι το background της πίστας.

A_MissionKills: Ο αριθμός των εχθρών που πρέπει ο παίκτης να καταρρίψει για να νικήσει στην πίστα.

A_UFOProperties: Ένα αντικείμενο τύπου EnemyProperties το οποίο έχει μέσα του όλα τα χαρακτηριστικά για τους εχθρούς τύπου “UFO”

A_AlienProperties: Ένα αντικείμενο τύπου EnemyProperties το οποίο έχει μέσα του όλα τα χαρακτηριστικά για τους εχθρούς τύπου “Aliens”

A_MeteoriteProperties: Ένα αντικείμενο τύπου EnemyProperties το οποίο έχει μέσα του όλα τα χαρακτηριστικά για τους εχθρούς τύπου “Meteorite”

Ο constructor αφού αρχικοποιεί σύμφωνα με τα ορίσματα κάποιες μεταβλητές καλεί την Init

Init

Η Init δημιουργεί τους εχθρούς της πίστας χρησιμοποιώντας πίνακες αντικειμένων και αρχικοποιεί κάποιες απαραίτητες μεταβλητές.

Ενδεικτικά η δημιουργία των εχθρών τύπου “Alien”:

```
If AlienProperties.Enabled Then
                For i As Integer = 0 To AlienProperties.EnemyCount -
1
                    AliensArray(i) = New
EnemyObject(AlienProperties.EnemyAnimation, AlienProperties.EnemyX,
AlienProperties.EnemyY, AlienProperties.EnemyMinX,
AlienProperties.EnemyMinY, AlienProperties.EnemyMaxX,
AlienProperties.EnemyMaxY, AlienProperties.EnemyMinSpeed,
AlienProperties.EnemyMaxSpeed, AlienProperties.EnemyAgressiveness,
AlienProperties.EnemyFireNum, AlienProperties.EnemyLaserDamage,
AlienProperties.EnemyCollisionDamage)
                    AliensArray(i).IsAlive = True
                Next i
```

```
End If
```

Στο τέλος η Init ορίζει τις μεταβλητές:

```
KillsCount = 0
MissionAccomplished = False
IntroSplashShown = False
IncreasedLevelOnce = False
```

Update

Η Update είναι η ρουτίνα η οποία καλείται συνεχώς από την timer1 και είναι υπεύθυνη για την κίνηση και την σχεδίαση όλων των εχθρών της πίστας.

Πιο αναλυτικά:

Η Update ξεκινάει με ένα block λειτουργιών το οποίο εκτελείται μόνο μια φορά και συγκεκριμένα την πρώτη φορά που θα τρέξει η Update από τον timer1. Με την χρήση μιας Boolean μεταβλητής, της InitedOnce κάνουμε αυτό το block να τρέξει μόνο μια φορά.

```
If Not InitedOnce Then

                CurrentBackground = LevelBackground

                Energy.IsAlive = False
                Player.Health = PlayerMaxHealth
                Player.ObjX = ((Form1.Width / 2) - (Player.ObjWidth /
2))
                Player.ObjY = Player.MaxY
                Player.ObjAnimation = AnimationStatus.Normal

                For i As Integer = 0 To PlayerLaserNum - 1
                    PlayerLaserShots(i).IsAlive = False
                Next

                LevelProgress = 0
                LevelProgressSTEP = (50 / MissionKills)
                InitedOnce = True
                Init()
End If
```

Μέσα στο μπλόκ της InitedOnce έχουμε κάποιες απαραίτητες αρχικοποιήσεις, όπως την αρχική θέση του παίκτη, την πρόοδο της πίστας κ.α.

Πέρα από την InitedOnce ξεκινάει το μέρος της Update το οποίο και εκτελείται κάθε φορά που καλείται η Update από το timer1. Αρχικά σχεδιάζουμε στο bmpDoubleBuffer το Background της πίστας. Είναι το πρώτο πράγμα που πρέπει να σχεδιαστεί έτσι ώστε όλα τα υπόλοιπα αντικείμενα να σχεδιαστούν πάνω από αυτό:

```
' Sxediasi tou background sto buffer
Dim g As Graphics = Graphics.FromImage(bmpDoubleBuffer)
g.DrawImage(LevelBackground, 0, 0)
```

Στην συνέχεια ακολουθεί ένα if το οποίο ελέγχει εάν έχει εμφανιστεί η όχι το εισαγωγικό splash screen της πίστας (το γραφικό το οποίο λέει π.χ. «Πίστα 1: Γαλαξίας Ανδρομέδας». Σε περίπτωση που έχει εμφανιστεί ξεκινάει το κυρίως μέρος της ρουτίνας το οποίο ελέγχει όλα τα αντικείμενα (εχθρούς UFO, Alien, Meteorites και το αντικείμενο bonus energy). Καλεί τις λειτουργίες ελέγχου κίνησης σχεδίασης και ελέγχου συγκρούσεων όπου αυτό απαιτείται, ενώ μετράει και αν κάποιος εχθρός έχει καταρριφτεί η όχι και αυξάνει το KillsCount. Ενδεικτικά:

```
If UFOProperties.Enabled Then

                    For i As Integer = 0 To UFOProperties.EnemyCount
- 1

                        UFOArray(i).CollisionCheck()
                        UFOArray(i).PositionUpdate()

                        If UFOArray(i).IsAlive = True Then
UFOArray(i).DrawObject()
                        If UFOArray(i).IsAlive = False And
UFOArray(i).MissilesActive = False Then KillsCount += 1
                        If UFOArray(i).IsAlive = False And Not
UFOArray(i).MissilesActive And Not MissionAccomplished Then
UFOArray(i).InitObject()

                    Next i

                End If
```

Αυτό είναι ένα απόσπασμα κώδικα που ελέγχει για το αν κάποιο UFO έχει χτυπηθεί από κάποιο Laser του παίκτη ή αν έχει συγκρουστεί με τον παίκτη (CollisionCheck), καλεί την λειτουργία ανανέωσης θέσης του UFO. Επίσης αυξάνει το KillsCount +1 εάν κάποιο από τα UFO έχει καταριφθεί και το ξαναενεργοποιεί για να εμφανιστεί ξανά εκτελώντας πάλι την InitObject. Επίσης στο τέλος γίνεται και ένας έλεγχος εάν η πίστα έχει ολοκληρωθεί ή όχι:

```
If KillsCount >= MissionKills Then MissionAccomplished = True
```

Εάν η πίστα έχει ολοκληρωθεί τότε:

```
' Mission Accomplished

                If MissionAccomplished Then

                    ShowGFXCentered(MissionAccomplishedBMP)

                    If Not IncreasedLevelOnce Then
                        IncreasedLevelOnce = True
                        FreezeGameForXSeconds(3)
                        CurrentLevel += 1
                        InitedOnce = False

                    End If
```

```
                End If
```

Σε αυτήν την περίπτωση εμφανίζεται το γραφικό το οποίο λέει ότι η πίστα ολοκληρώθηκε και έπειτα αυξάνει μόνο μια φορά κατά 1 (με την βοήθεια της IncreasedLevelOnce) τον μετρητή της τρέχουσας πίστας.

2.7.6 Η κλάση PlayerObject

Η κλάση PlayerObject είναι η κλάση η οποία υλοποιεί τον έλεγχο και την κίνηση του διαστημόπλοιου του παίκτη. Στο παιχνίδι το αντικείμενο τύπου PlayerObject είναι το Player. Η PlayerObject κληρονομεί τα χαρακτηριστικά της και τις λειτουργίες τις από την GameObjectAnimated.

Μεταβλητές

```
Public MaxHealth As Integer = PlayerMaxHealth
Public Health As Integer = MaxHealth
Dim UpFlag As Boolean = False
Dim DownFlag As Boolean = False
Dim LeftFlag As Boolean = False
Dim RightFlag As Boolean = False
```

Η MaxHealth είναι η μεταβλητή που αναφέρεται στην μέγιστη ενέργεια του παίκτη και παίρνει την τιμή της από την σταθερά PlayerMaxHealth. Η Health είναι η τρέχουσα ενέργεια (ζωή) του παίκτη. Οι υπόλοιπες Boolean χρησιμοποιούνται όπως μαρτυράνε οι ονομασίες τους για τον έλεγχο της κίνησης του διαστημόπλοιου και η ανάθεση τιμών σε αυτές έχουν να κάνουν με το τι πατιέται στο πληκτρολόγιο και συγκεκριμένα με την τιμή της shared μεταβλητής KeyFlag.

Constructor

Ο constructor της PlayerObject δεν έχει κάτι το διαφορετικό με αυτόν της GameObject και απλώς καλεί τον constructor της GameObject ενώ στη συνέχεια θέτει ως true το IsAlive του αντικειμένου:

```
MyBase.New(AnObjImageArray, AnObjX, AnObjY, AnMinX, AnMinY, AnMaxX,
AnMaxY, AnObjMinSpeed, AnObjMaxSpeed)
IsAlive = True
```

PositionUpdate

Η PositionUpdate του PlayerObject διαβάζει την τιμή της shared μεταβλητής KeyFlag και αναλόγως αναθέτει τιμές True η False στις μεταβλητές UpFlag, DownFlag, LeftFlag, RightFlag. Ενδεικτικά:

```
Select Case (KeyFlag)
```

```
                Case Directions.GoUp
                    UpFlag = True
                    DownFlag = False
                    LeftFlag = False
                    RightFlag = False
```

Μετά από την Case γίνεται η κίνηση του αντικειμένου ανάλογα με τις τιμές που έχουν δοθεί:

```
If UpFlag Then ObjY -= ObjSpeed
If DownFlag Then ObjY += ObjSpeed
If LeftFlag Then ObjX -= ObjSpeed
If RightFlag Then ObjX += ObjSpeed
```

Στο τέλος της, η PositionUpdate εκτελεί την λειτουργία BoundCheck η οποία δεν υλοποιείται εκ νέου, αλλά εκτελείται από την μητρική της κλάση GameObject και που απλά περιορίζει την κίνηση του αντικειμένου μέσα στα δοσμένα όρια.

PlayerStatus

Η PlayerStatus ασχολείται με την σχεδίαση της μπάρας κατάστασης που βρίσκεται στο κάτω μέρος της οθόνης και που έχει πληροφορίες για την ενέργεια του παίκτη αλλά και για την πρόοδό του στην πίστα.

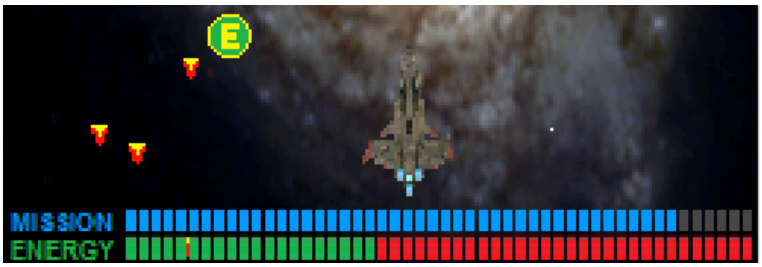
Ενδεικτικά:

```
For i As Integer = 1 To 50

                dstRect = New Rectangle(((BlueBar.Width + 1) * (i)) +
EnergyTextBMP.Width, Form1.Height - (BlueBar.Height * 2) - 1,
BlueBar.Width, 9)

                If i > LevelProgress Then
                    g.DrawImage(GreyBar, dstRect, 0, 0,
GreyBar.Width, GreyBar.Height, GraphicsUnit.Pixel, TransparencyAttr)
                Else
                    g.DrawImage(BlueBar, dstRect, 0, 0,
BlueBar.Width, BlueBar.Height, GraphicsUnit.Pixel, TransparencyAttr)
                End If
Next i
```

Η ρουτίνα αυτή σχεδιάζει τα μικρά τετράγωνα που δείχνουν την πρόοδο του παίκτη . Σχεδιάζονται 50 τετράγωνα στις σωστές θέσεις ένα προς ένα, επιλέγοντας μπλέ όσο ο μετρητής i είναι μικρότερος από LevelProgress και από εκεί και πέρα μέχρι το 50 γκρί. Έτσι δημιουργείται η μπάρα προόδου της πίστας. Αντίστοιχη λειτουργία συμβαίνει και για την ένδειξη της ενέργειας του παίκτη.



Επίσης πέρα από την σχεδίαση των γραφικών της μπάρας κατάστασης, γίνεται έλεγχος για το αν η ενέργεια (υγεία) του παίκτη μηδενιστεί, οπότε και ορίζει την κατάστασή του παίκτη σε “Explosion”:

```
If Health <= 0 Then
                ObjAnimation = AnimationStatus.Explosion
End If
```

2.7.7 Η κλάση EnemyMissile

Η κλάση EnemyMissile αφορά τα laser των εχθρών και κληρονομεί τα χαρακτηριστικά της από την GameObject. Οι διαφορές της δεν είναι πολλές σε σχέση με την GameObject. Μια προσθήκη είναι μια μεταβλητή η DamageFromEnemyLaser τύπου integer και η οποία λέει πόσο ζημιά προκαλεί στον παίκτη το laser του εχθρού. Η DamageFromEnemyLaser παρουσιάζεται και σαν όρισμα στον Constructor που έχει αυτό το όρισμα παραπάνω σε σχέση με την GameObject.

PositionUpdate

Η PositionUpdate είναι εξαιρετικά απλή, καθώς τα laser του εχθρού απλώς κινούνται προς τα κάτω:

```
Public Overrides Sub PositionUpdate()
            ObjY += ObjSpeed
            BoundCheck()
End Sub
```

BoundCheck

Η BoundCheck είναι το πιο ενδιαφέρον κομμάτι στην EnemyMissile. Θέτει το EnemyMissile αντικείμενο false όταν αυτό υπερβεί το MaxY του (δηλαδή όταν το laser φύγει από το κάτω μέρος της οθόνης, που είναι και το MaxY του) ενώ υλοποιεί και τον έλεγχο σύγκρουσης με το διαστημόπλοιο του παίκτη με τον εξής κώδικα:

```
If ((Player.ObjX + Player.ObjWidth > ObjX) And (Player.ObjX < ObjX +
ObjWidth)) And ((Player.ObjY + Player.ObjHeight > ObjY) And
(Player.ObjY < ObjY + ObjHeight)) Then

                Player.Health -= DamageFromEnemyLaser
                Player.ObjAnimation = AnimationStatus.Damage
```

```
                IsAlive = False


End If
```

Η If συνθήκη γίνεται true στην περίπτωση που το laser συγκρουστεί με τον παίκτη και τότε μειώνεται η ενέργεια του παίκτη κατά DamageFromEnemyLaser μονάδες, το animation του παίκτη ορίζεται ως “Damage” και τέλος απενεργοποιείται το laser το οποίο χτύπησε τον παίκτη.

2.7.9 Η κλάση PlayerMissile

Η PlayerMissile αφορά τα laser του παίκτη. Κληρονομεί τα χαρακτηριστικά της από την GameObject, χωρίς καμμία αλλαγή στον constructor της που απλά καλεί τον constructor της GameObject:

```
MyBase.New(AnObjImage, AnObjX, AnObjY, AnMinX, AnMinY, AnMaxX,
AnMaxY, AnObjMinSpeed, AnObjMaxSpeed)
```

PositionUpdate, BoundCheck

Φυσικά υλοποιείται η PositionUpdate όπως υποχρεούται από την GameObject, με τον εξής κώδικα:

```
If IsAlive = True Then ObjY -= ObjSpeed

If IsAlive = False And KeyFlag = Directions.Fire Then

                IsAlive = True
                ObjX = Form1.Player.ObjX +
((Form1.Player.ObjImageArray(0).Width) / 2) - 2
                ObjY = Form1.Player.ObjY - ObjImage.Height + ObjSpeed
                KeyFlag = Directions.NullDirection_FireException


End If

BoundCheck()
```

Εάν το laser του παίκτη είναι ενεργό, τότε απλά αυτό κινείται στον άξονα Y προς τα πάνω κατά ObjSpeed pixels. Εάν το laser δεν είναι ενεργό και έχει πατηθεί το πλήκτρο που αντιστοιχεί στον πυροβολισμό τότε ενεργοποιείται, σε σημείο ObjX και ObjY που αντιστοιχεί ακριβώς μπροστά στην «μύτη» του διαστημόπλοιου του παίκτη. Στην συνέχεια το KeyFlag παίρνει την τιμή Directions.NullDirection_FireException αποτρέπωντας τον συνεχή πυροβολισμό και των υπόλοιπων laser. Πριν ολοκληρωθεί, η PositionUpdate καλεί την BoundCheck η οποία πολύ απλά απενεργοποιεί το laser όταν αυτό περάσει το πάνω οριακό μέρος της οθόνης:

```
Public Overrides Sub BoundCheck()

            If ObjY <= MinY Then IsAlive = False

End Sub
```

2.7.10 Η κλάση StarObject

Η StarObject κληρονομεί τα χαρακτηριστικά της από την GameObject και χρησιμοποιείται για τα αστέρια, μικρά και μεγάλα του background. Η διαφορά με την GameObject είναι οι random τιμές που δίνονται στον constructor για τα αρχικά X,Y αλλά και την random ταχύτητα η οποία κυμαίνεται στο εύρος AnObjMinSpeed, AnObjMaxSpeed.

PositionUpdate

Η PositionUpdate είναι απλή. Το αντικείμενο κινείται κατά +ObjSpeed pixels κάθετα. Έπειτα καλείται η BoundCheck

BoundCheck

Εάν το αντικείμενο ξεπεράσει το MaxY, ξαναεμφανίζεται στο πάνω μέρος της οθόνης (Y=0), σε τυχαίο X, με τυχαία ταχύτητα πάλι στο εύρος ObjMinSpeed, ObjMaxSpeed.

2.7.11 Η κλάση EnemyObject

Η κλάση EnemyObject κληρονομεί τα χαρακτηριστικά της από την GameObjectAnimated και χρησιμοποιείται για τους εχθρούς, τύπου UFO και Alien.

Μεταβλητές

Πέρα από αυτές της GameObjectAnimated, η EnemyObject χρησιμοποιεί τις εξής μεταβλητές:

```
Dim AlienLaserShots(MAXIMUMLASERPERTYPE) As EnemyMissile
Dim AlienLaserBMP As Bitmap =
Form1.BitmapStreamLoader("AlienLaser0.bmp")  'Alien's Laser bitmap
Dim KillCounted As Boolean = False
Dim CollisionCounted As Boolean = False
Dim ObjAggressiveness, ObjLaserNum As Integer
Public DamageFromLaser, DamageFromCollision As Integer
```

Ο AlienLaserShots είναι ένας πίνακας αντικειμένων τύπου EnemyMissile και χρησιμοποιείται για τα laser του εχθρού. Η AlienLaserBMP είναι το γραφικό του laser. Η KillCounted είναι μια boolean η οποία λέει εάν η κατάριψη του εχθρικού αντικειμένου από τον παίκτη έχει προσμετρηθεί η όχι. Η CollisionCounted είναι μια boolean που λέει εάν η

σύγκρουση του αντικειμένου με το διαστημόπλοιο του παίκτη έχει μετρηθεί η όχι. Η ObjAgressiveness ανφέρεται στην επιθετικότητα του αντικειμένου και παίρνει την τιμή της από το αντίστοιχο όρισμα στον constructor. Παρομοίως και για την ObjLaserNum που αναφέρεται στον αριθμό των laser του εχθρού που μπορούν να υπάρχουν ταυτόχρονα στην οθόνη. Η DamageFromLaser και η DamageFromCollision δείχνουν τι ζημιά παθαίνει ο παίκτης όταν χτυπηθεί από το laser του εχθρού ή όταν συγκρουστεί με αυτόν αντίστοιχα.

Constructor

Ο Constructor της κλάσης αυτής σε σχέση με τον constructor της GameObjectAnimated παίρνει τα εξής παραπάνω ορίσματα:

```
A_Agressiveness As Integer, ByVal A_LaserNum As Integer, ByVal
A_DamageFromLaser As Integer, ByVal A_DamageFromCollision As Integer
```

Τα παραπάνω ορίσματα αναφέρονται αντίστοιχα στην επιθετικότητα του αντικειμένου, στον αριθμό των laser του, στην ζημιά που προκαλεί στον παίκτη με τα laser του και τέλος με σύγκρουση.

Ο cunstroctor κάνει τις απαραίτητες αναθέσεις τιμών σε κάποιες μεταβλητές και έπειτα καλεί την InitObject

InitObject

Η InitObject αρχικοποιεί διάφορες μεταβλητές ενώ κατασκευάζει και τον πίνακα αντικειμένων τύπου EnemyMissile ο οποίος είναι τα laser του εχθρού, χρησιμοποιώντας τα κατάλληλα ορίσματα.

```
For i As Integer = 0 To ObjLaserNum - 1
                AlienLaserShots(i) = New EnemyMissile(AlienLaserBMP,
0, 0, 0, 0, Form1.Width, Form1.Height - AlienLaserBMP.Height, 5, 5,
DamageFromLaser)
Next i
```

PositionUpdate

Η PositionUpdate αρχικά υλοποιεί ένα απλό μοντέλο κίνησης του εχθρού το οποίο είναι απλώς μια bouncing κίνηση μέσα στον επιτρεπτό χώρο κίνησης.

Ο τρόπος κίνησης των εχθρών (bouncing)

Ο κώδικας που υλοποιεί αυτή την κίνηση είναι ο εξής:

```
If (ObjX + ObjStepX) > MaxX Then ObjStepX = -(ObjSpeed)
If (ObjY + ObjStepY) > MaxY Then ObjStepY = -(ObjSpeed)

If ObjX <= MinX Then ObjStepX = ObjSpeed
If ObjY <= MinY Then ObjStepY = ObjSpeed

ObjX += ObjStepX
ObjY += ObjStepY
```

Στην συνέχεια υλοποιείται η λειτουργία όπου ο εχθρός αποφασίζει εάν θα πυροβολήσει η όχι. Εδώ για κάθε διαθέσιμο laser που έχει ο εχθρός αποφασίζει με βάση την επιθετικότητά του (ObjAgressiveness) εάν θα πυροβολήσει η όχι. Εάν πυροβολήσει δίνεται η αρχική θέση X,Y για το laser που είναι μπροστά στην «μύτη» του εχθρού, ενώ ενεργοποιείται και το boolean MissilesActive που σηματοδοτεί ότι υπάρχουν ενεργά laser του εχθρού στην οθόνη:

```
For i = 0 To ObjLaserNum - 1

                    If (ObjRandomizer(0, 1000) < ObjAggressiveness)
And AlienLaserShots(i).IsAlive = False Then

                        AlienLaserShots(i).ObjX = ObjX + (ObjWidth /
2)
                        AlienLaserShots(i).ObjY = ObjY + ObjHeight
                        AlienLaserShots(i).IsAlive = True
                        MissilesActive = True
```

```
                    End If
Next
```

Η PositionUpdate της κλάσης EnemyObject κλείνει με την ρουτίνα που υλοποιεί την ανανέωση της θέσης των laser και των σχεδιασμό αυτών:

```
AMissileAlive = False
            If MissilesActive = True Then
                For i = 0 To ObjLaserNum - 1
                    If AlienLaserShots(i).IsAlive = True Then
                        AlienLaserShots(i).PositionUpdate()
                        AlienLaserShots(i).DrawObject()
                        AMissileAlive = True
                    End If

                Next
```

Η ρουτίνα εκτελείται μόνο στην περίπτωση που υπάρχει κάποιο ενεργό laser στην οθόνη και καλούνται διαδοχικά η PositionUpdate και η DrawObject του AlienMissile στοιχείου i του πίνακα AlienLaserShots.

CollisionCheck

Στην CollisionCheck της κλάσης EnemyObject γίνονται οι απαραίτητοι έλεγχοι σύγκρουσης μεταξύ του εχθρικού αντικειμένου και των laser του παίκτη, ενώ εδώ υπολογίζεται εάν θα εμφανιστεί bonus ενέργειας κατά την καταστροφή του εχθρού.

Ενδεικτικά:

```
If ((PlayerLaserShots(i).ObjX + PlayerLaserShots(i).ObjWidth > ObjX)
And (PlayerLaserShots(i).ObjX < ObjX + ObjWidth)) And
((PlayerLaserShots(i).ObjY + PlayerLaserShots(i).ObjHeight > ObjY)
And (PlayerLaserShots(i).ObjY < ObjY + ObjHeight)) Then

      ObjAnimation = AnimationStatus.Explosion
      If KillCounted = False Then
            LevelProgress += LevelProgressSTEP
            KillCounted = True

      End If
```

Στο παραπάνω κομμάτι κώδικα γίνεται ο έλεγχος για το εάν κάποιο από τα laser του παίκτη έρθει σε επαφή με τον εχθρό. Σε αυτήν την περίπτωση, το animation του εχθρού ορίζεται σε κατάσταση έκρηξης, ενώ αυξάνεται και η πρόοδος της πίστας.

```
If (ObjRandomizer(0, 100) < EnergyPossibility) And Energy.IsAlive =
False Then

      Energy.ObjX = ObjX
      Energy.ObjY = ObjY
      Energy.IsAlive = True

End If
```

Στο παραπάνω κομμάτι κώδικα το οποίο εκτελείται όταν ένας εχθρός χτυπηθεί από το laser του παίκτη, εμφανίζεται όταν αυτό προκύψει τυχαία, το αντικείμενο Energy το οποίο είναι τύπου BonusObject, ακριβώς στη θέση του εχθρικού αντικειμένου.

2.7.12 Η κλάση MeteoriteObject

Η κλάση MeteoriteObject κληρονομεί τα χαρακτηριστικά της από την GameObjectAnimated και χρησιμοποιείται για τους μετεωρίτες. Υλοποιεί κίνηση των μετεωριτών από το πάνω μέρος της οθόνης προς τα κάτω, ενώ κάνει έλεγχο σύγκρουσης μετεωρίτη με διαστημόπλοιο του παίκτη.

Constructor

Το πρόσθετο χαρακτηριστικό σε σχέση με την GameObjectAnimated όσον αφορά τον constructor είναι το όρισμα A_DamageFromCollision το οποίο αναφέρεται στο πόσο ζημιά προκαλεί η σύγκρουση του αντικειμένου με τον παίκτη.

InitObject

Η InitObject αρχικοποιεί τις μεταβλητές θέσης και ταχύτητας του αντικειμένου. Ενδεικτικά η τυχαία κατεύθυνση του αντικειμένου στον X άξονα:

```
If Int(ObjRandomizer(0, 1)) = 1 Then ObjStepX = -ObjStepX
```

PositionUpdate

Η PositionUpdate είναι αρκετά απλή, καθώς το αντικείμενο απλά κινείται προς μια κατεύθυνση:

```
ObjY += ObjStepY
ObjX += ObjStepX

BoundCheck()
```

BoundCheck

Η BoundCheck ελέγχει εάν το αντικείμενο έχει φύγει εκτός οθόνης και σε αυτήν την περίπτωση το ξαναεμφανίζει με νέα τυχαία χαρακτηριστικά πάλι στο πάνω μέρος της οθόνης

CollisionCheck

Η CollisionCheck έχει υλοποιηθεί με παρόμοιο τρόπο όπως και η αντίστοιχη της κλάσης EnemyObject. Ελέγχει για το εάν ο μετεωρίτης έχει κτυπηθεί από κάποιο laser του παίκτη αλλά και για το εάν έχει συγκρουστεί με αυτόν.

2.7.13 Η κλάση BonusObject

Η κλάση BonusObject κληρονομεί τα χαρακτηριστικά της από την GameObjectAnimated και χρησιμοποιείται για το bonus ενέργειας του παίκτη το οποίο έχει πιθανότητα να εμφανιστεί κατά την κατάρριψη κάποιου εχθρού. Υλοποιεί μια απλή κίνηση του αντικειμένου προς τα κάτω, ενώ ελέγχει και για το αν ο παίκτης ήρθε σε επαφή με αυτό.

Constructor

Ο constructor που χρησιμοποιείται είναι αυτός της μητρικής κλάσης GameObjectAnimated.

InitObject

Αρχικοποίηση μεταβλητών.

PositionUpdate

Απλά κίνηση προς τα κάτω:

```
If IsAlive = True Then

                ObjY += ObjStepY
                BoundCheck()

End If
```

BoundCheck

Απενεργοποιεί το αντικείμενο όταν αυτό βγεί εκτός οθόνης.

CollisionCheck

Η CollisionCheck ελέγχει εάν ο παίκτης ακούμπησε το bonus και κάνει τις απαραίτητες ενέργειες: Προσθέτει ενέργεια στον παίκτη και εξαφανίζει το bonus αντικείμενο. Επίσης κάνει την κατάσταση του παίκτη Damage καθαρά για λόγους αισθητικής για να φανεί μια αντίδραση στο διαστημόπλοιο του παίκτη καθώς ακουμπάει το bonus.

```
Public Sub CollisionCheck()

            If ((Player.ObjX + Player.ObjWidth > ObjX) And
(Player.ObjX < ObjX + ObjWidth)) And ((Player.ObjY + Player.ObjHeight
> ObjY) And (Player.ObjY < ObjY + ObjHeight)) Then
                Player.Health += EnergyBonus
                If Player.Health > Player.MaxHealth Then
Player.Health = Player.MaxHealth
                Player.ObjAnimation = AnimationStatus.Damage
                IsAlive = False
            End If

End Sub
```

3. Παραμετροποίηση

Η παραμετροποίηση του παιχνιδιού είναι πλέον σχετικά εύκολη από τον προγραμματιστή, καθώς μπορεί με απλή ανάθεση τιμών να δημιουργήσει νέες πίστες, χωρίς να χρειαστεί να αλλάξει κάποιο κομμάτι κώδικα. Η ίδια δυνατότητα μπορεί να δοθεί και στον απλό χρήστη μέσα από ένα εύχρηστο πάνελ με γνώριμα για αυτόν αντικείμενα (κουμπιά, sliders, checkboxes) τα οποία παραμετροποιούν κάποια επιλεγμένα χαρακτηριστικά των εχθρών και της πίστας.

3.1 Παράδειγμα κατασκευής πίστας από τον προγραμματιστή

Ο προγραμματισμός της πίστας γίνεται περνώντας τα κατάλληλα ορίσματα τύπου EnemyProperties στον constructor ενός αντικειμένου τύπου GameLevel. Η GameLevel, η οποία είναι η κλάση που υλοποιεί μια πίστα παιχνιδιού, έχει την δυνατότητα παραμετροποίησης:

- του γραφικού που θα φανεί στην αρχή της πίστας (π.χ. ένα bitmap που θα λέει «Δοκιμαστική Πίστα! Καλή επιτυχία»
- του γραφικού που θα έχει η πίστα ως background
- τον αριθμό των εχθρών που πρέπει να καταρρίψει ο παίκτης για να τελειώσει η πίστα
- τις ιδιότητες 3 τύπων εχθρών. Τύπου «Alien», τύπου «UFO» και μετεωριτών.

```
Public Sub New(ByVal A_IntroSplash As Bitmap, ByVal A_LevelBackground
As Bitmap, ByVal A_MissionKills As Integer, ByVal A_UFOProperties As
EnemyProperties, ByVal A_AlienProperties As EnemyProperties, ByVal
A_MeteoriteProperties As EnemyProperties)
```

Για παράδειγμα, εάν θέλουμε μια πίστα όπου να υπάρχουν **πολλοί** εχθροί τύπου «Alien», οι οποίοι να είναι **γρήγοροι**, να είναι **πολύ επιθετικοί** αλλά τα όπλα τους να **μην είναι και τόσο δυνατά**, μπορούμε να χρησιμοποιήσουμε το εξής:

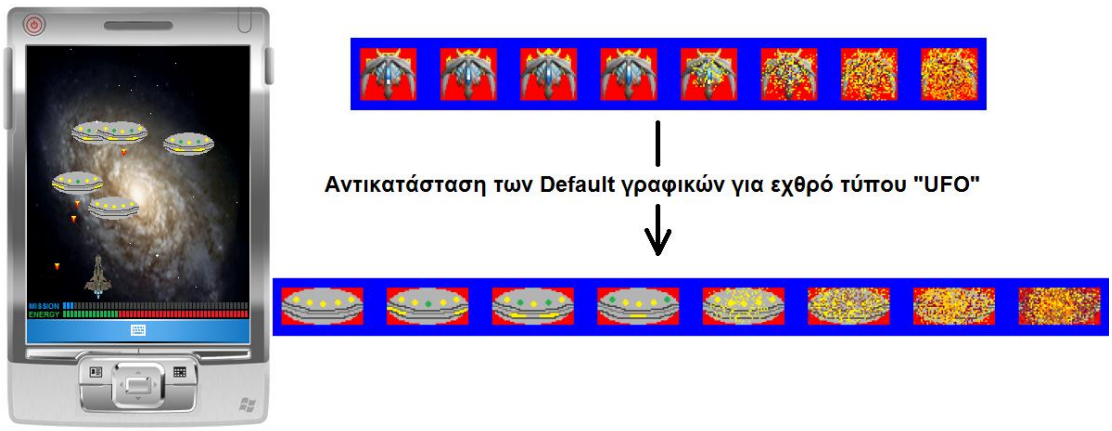
```
DefaultAlienProperties.Enabled = True
DefaultAlienProperties.EnemyAgressiveness = 20
DefaultAlienProperties.EnemyAnimation = AlienAnimated
DefaultAlienProperties.EnemyCollisionDamage = 10
DefaultAlienProperties.EnemyCount = 10
DefaultAlienProperties.EnemyFireNum = 2
DefaultAlienProperties.EnemyLaserDamage = 1
DefaultAlienProperties.EnemyMaxSpeed = 10
DefaultAlienProperties.EnemyMaxX = Me.Width
DefaultAlienProperties.EnemyMaxY = Me.Height / 2 +
AlienAnimated(0).Height
```

```
DefaultAlienProperties.EnemyMinSpeed = 8
DefaultAlienProperties.EnemyMinX = 0
DefaultAlienProperties.EnemyMinY = 0
DefaultAlienProperties.EnemyX = 0
DefaultAlienProperties.EnemyY = 0
```

Το EnemyAgressiveness = 20 κάνει τον εχθρό ιδιαίτερα επιθετικό, το EnemyCount = 10 ορίζει 10 εχθρούς ταυτόχρονα στην οθόνη, το LaserDamage = 1 ορίζει ότι ο παίκτης θα χάνει μόνο 1 μονάδα ενέργειας αν χτυπηθεί από το laser του εχθρού. Τα EnemyMinSpeed = 8 και EnemyMaxSpeed = 10 ορίζουν την ελάχιστη και μέγιστη δυνατή ταχύτητα του εχθρού η οποία είναι από 8 έως 10 pixel κάθε φορά, κάτι που καθιστά τους εχθρούς ιδιαίτερα γρήγορους.

Ο προγραμματιστής μπορεί εάν θέλει να αλλάξει ακόμα και τα γραφικά των εχθρών, αρκεί να φορτώσει τις απαραίτητες 4 εικόνες της Normal κατάστασης αλλά και 4 εικόνες της έκρηξης ως «Embedded Resource» στο Visual Studio και να τις φορτώσει σε ένα bitmap array το οποίο και θα αναθέσει στην EnemyAnimation.

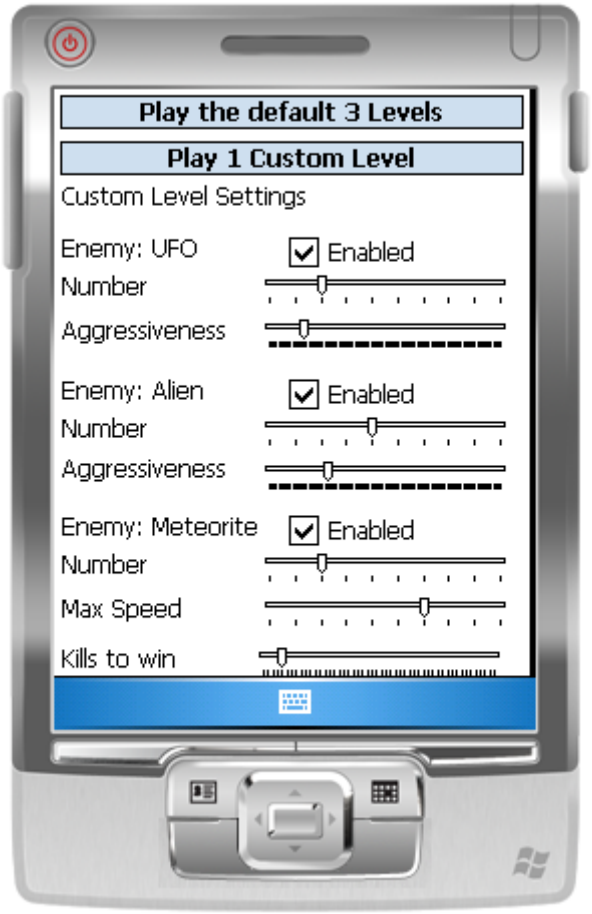
```
DefaultAlienProperties.EnemyAnimation = My_Custom_Animation
```



Παρομοίως θα προγραμματιστούν και οι εχθροί τύπο «UFO» αλλά και οι μετεωρίτες. Εν τέλει θα περάσουμε αυτά τα ορίσματα στον constructor του αντικειμένου τύπου GameLevel. Υπενθυμίζεται ότι εκεί έχουμε την δυνατότητα να ορίσουμε και το πόσοι εχθροί πρέπει να καταρριφθούν από τον παίκτη για να τελειώσει η πίστα αλλά και το γραφικό background.

3.2 Custom πίστα από τον χρήστη

Ο παίκτης έχει την δυνατότητα κατασκευής μιας πίστας από το αρχικό μενού, χρησιμοποιώντας γνώριμα checkboxes και sliders.



Ο χρήστης μπορεί να επιλέξει να παίξει τις 3 προκατασκευασμένες πίστες, ή να παίξει αυτήν που ορίζεται από αυτόν πατώντας το αντίστοιχο button. Για τον εχθρό τύπου «UFO» και για τον εχθρό τύπου «Alien» μπορεί να επιλέξει εάν θα υπάρχουν ή όχι (Enabled), τον αριθμό αυτών (Number) αλλά και την επιθετικότητά τους (Agressiveness). Παρομοίως και για τους μετεωρίτες, με μόνη διαφορά αντί για την επιθετικότητα η επιλογή (Max Speed) ανφέρεται στην μέγιστη ταχύτητα αυτών. Το τελευταίο slider (Kills to win) είναι ο αριθμός των εχθρών που πρέπει να καταρρίψει ο παίκτης για να κερδίσει την πίστα.

4. Πηγές

Σημαντική ήταν η τεχνική βοήθεια που προήλθε μετά από έρευνα στο διαδίκτυο.

www.Codeproject.com

www.Vbdotnetheaven.com

www.Developerfusion.com

www.Gamedev.net

www.Xtremevbtalk.com

www.msdn.Microsoft.com

www.Geekswithblogs.net

www.Planet-source-code.com

www.Vbgames.com

www.vbprogramming.8k.com

www.Caspercomsci.com

www.Devx.com