

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΡΗΤΗΣ ΤΜΗΜΑ ΕΠΙΣΤΗΜΗΣ ΥΠΟΛΟΓΙΣΤΩΝ

# ΑΝΑΠΤΥΞΗ ΕΞΥΠΝΩΝ ΔΙΕΠΑΦΩΝ ΚΑΙ ΠΑΙΧΝΙΔΙΩΝ

---

## ΕΡΓΑΣΤΗΡΙΑΚΗ ΑΣΚΗΣΗ - PROJECT

ΔΙΔΑΣΚΩΝ: ΑΝΤΩΝΙΟΣ ΣΑΒΒΙΔΗΣ

Χειμερινό εξάμηνο 2015 - 2016



# Περιεχόμενα

<u>Εισαγωγή.....</u>	<u>3</u>
<u>Ενότητα 1: Βασικό Gameplay.....</u>	<u>3</u>
<u>Γενική περιγραφή.....</u>	<u>3</u>
<u>Χειρισμός.....</u>	<u>3</u>
<u>Χαρακτήρες του παιχνιδιού.....</u>	<u>3</u>
<u>Super Ace.....</u>	<u>3</u>
<u>Μικρά εχθρικά αεροσκάφη.....</u>	<u>3</u>
<u>Μεσαία εχθρικά αεροσκάφη.....</u>	<u>3</u>
<u>Μεγάλα εχθρικά αεροσκάφη.....</u>	<u>4</u>
<u>Power-ups.....</u>	<u>4</u>
<u>Η οθόνη του παιχνιδιού.....</u>	<u>4</u>
<u>Αρχή και τέλος πίστας.....</u>	<u>4</u>
<u>Ενότητα 2: Κατασκευαστικές οδηγίες.....</u>	<u>5</u>
<u>Έλεγχος χρονισμού της δράσης .....</u>	<u>5</u>
<u>Κατασκευή της πίστας.....</u>	<u>6</u>
<u>Κίνηση εχθρικών αεροσκαφών.....</u>	<u>7</u>
<u>Side fighters.....</u>	<u>9</u>
<u>Προσαρμογές (configuration facilities).....</u>	<u>13</u>
<u>Pause – Resume.....</u>	<u>13</u>
<u>Ομαδική εργασία και παράδοση.....</u>	<u>13</u>
<u>Αναφορές.....</u>	<u>13</u>

## Εισαγωγή

---

Σε αυτήν την εργασία θα υλοποιήσετε το κλασσικό παιχνίδι 1942. Ένα χαρακτηριστικό βίντεο με gameplay από την arcade έκδοση του παιχνιδιού βρίσκεται [εδώ](#) και μπορείτε να παίξετε μια έκδοση του παιχνιδιού [εδώ](#).

Ζητούμενο της εργασίας είναι η υλοποίηση **τουλάχιστον** μιας πίστας που να ικανοποιεί το βασικό gameplay, να περιλαμβάνει **στο ελάχιστο** τους **εχθρούς** και τα **power-ups** που θα οριστούν [παρακάτω](#).

Το υπόλοιπο έγγραφο περιλαμβάνει μια περιγραφή του [βασικού gameplay](#), [κατασκευαστικές οδηγίες](#) για τα επιμέρους κομμάτια του παιχνιδιού, [οδηγίες για την παράδοση](#) και χρήσιμα [links / αναφορές](#) για το 1942.



Ιδέα! Στη δικιά σας υλοποίηση μπορείτε να κρατήσετε το βασικό gameplay ίδιο και να παίξετε με διάφορες άλλες παραμέτρους ώστε να δώσετε τη δική σας πινελιά στο παιχνίδι.

Πιθανά στοιχεία του παιχνιδιού με τα οποία μπορείτε να πειραματιστείτε είναι:

- Διαφορετικά sprites και background image
- Έξτρα πίστες
- Νέα power-ups
- Νέοι εχθροί
- Cheat codes
- Υποστήριξη για 2<sup>ο</sup> παίκτη (είτε παίζοντας εναλλάξ είτε παράλληλα)

## Ενότητα 1: Βασικό Gameplay

---

### Γενική περιγραφή

Το 1942 δημιουργήθηκε το 1984 από την Capcom. Σκοπός του παιχνιδιού είναι ο παίκτης να καταστρέψει τον αντίπαλο στόλο αεροσκαφών. Ο παίκτης χειρίζεται το "Super Ace", ένα αεροσκάφος τύπου [Lockheed P-38 Lightning](#) και καταρρίπτει τα εχθρικά αεροσκάφη (διαφόρων τύπων όπως [Kawasaki Ki-61s](#), [Mitsubishi A6M Zeros](#), και [Kawasaki Ki-48s](#)) αποφεύγοντας τα πυρά τους. Υπάρχουν 32 πίστες στο σύνολο και το παιχνίδι ξεκινάει από την πίστα 32 και συνεχίζει με αντίστροφη σειρά μέχρι την πίστα 1. Σε κάποιες πίστες υπάρχουν και τελικοί αρχηγοί (boss fights) που είναι αεροσκάφη τύπου [Nakajima G8N](#).

Στην δικιά σας υλοποίηση θα υλοποιήσετε κατ' ελάχιστον μια πίστα, χωρίς απαραίτητα να υλοποιήσετε τελικό αρχηγό.

### Χειρισμός



: Το Super Ace κινείται προς όλες τις κατευθύνσεις (αριστερά-δεξιά και πάνω-κάτω).



: Το Super Ace πυροβολεί εκτοξεύοντας δύο σφαίρες (ή τέσσερις αν έχει πάρει το κατάλληλο power-up) προς τα πάνω.



: Ο χαρακτήρας εκτελεί ένα ελιγμό αποφυγής 360 μοιρών (loop) κατά τη διάρκεια του οποίου δε μπορεί να χτυπηθεί από εχθρικά αεροσκάφη ή πυρά. Υπάρχει συγκεκριμένος αριθμός ελιγμών που μπορεί να εκτελεστεί, με τον αριθμό αυτό να εμφανίζεται κάτω δεξιά στην οθόνη.

## Χαρακτήρες του παιχνιδιού

### Super Ace



*Super Ace* είναι το αεροπλάνο που χειρίζεται ο παίκτης. Μπορεί να κινείται ελεύθερα στη οθόνη, τόσο αριστερά-δεξιά όσο και πάνω κάτω (χωρίς όμως να αλλάζει το animation film του για την κίνηση προς τα κάτω). Μπορεί να εκτοξεύει σφαίρες προς τα πάνω και να κάνει ένα ελιγμό αποφυγής (περιορισμένες όμως φορές). Αν χτυπηθεί από εχθρική σφαίρα ή συγκρουστεί με εχθρικό αεροσκάφος καταρρίπτεται και ο παίκτης χάνει μια ζωή.

### Μικρά εχθρικά αεροσκάφη

Μονοκινητήρια αεροσκάφη	Δικινητήρια αεροσκάφη	Jet αεροσκάφη

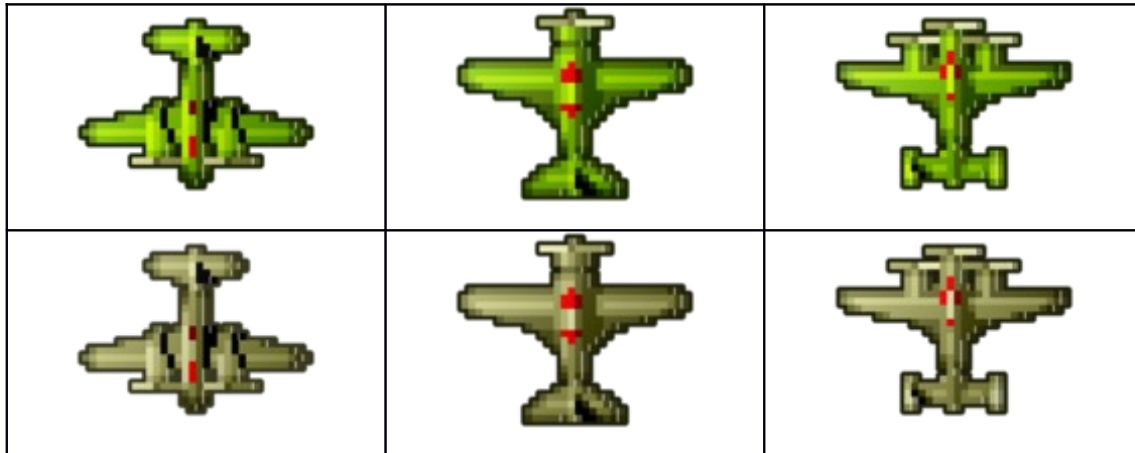
Αποτελούν την πλειοψηφία των αεροσκαφών που συναντάμε στο παιχνίδι. Οι επιμέρους κινήσεις τους διαφέρουν λίγο, αλλά γενικά μπαίνουν στη σκηνή, πυροβολούν μια-δυο φορές και κατόπιν φεύγουν. Ακολουθούν προδιαγεγραμμένη πορεία και αν συγκρουστούν με τον παίκτη τον καταρρίπτουν. Ο παίκτης μπορεί να τα καταρρίψει με ένα χτύπημα. Τα γκρι αεροσκάφη έρχονται από το πάνω μέρος της οθόνης ενώ τα πράσινα έρχονται από τα πλάγια.

Τα μονοκινητήρια αεροσκάφη που έρχονται από τα πλάγια κάνουν τυχαίους κύκλους πριν φύγουν. Αυτά που έρχονται από το πάνω μέρος της οθόνης μπορεί να έρθουν τελείως κάθετα και όταν φτάσουν τον παίκτη στρίβουν προς αυτόν, αλλιώς μπορεί να έρθουν υπό γωνία και όταν φτάσουν τον παίκτη κάνουν τούμπα και γυρνάνε πίσω απομακρυνόμενα από αυτόν.

Τα δικινητήρια αεροσκάφη απλά μένουν περισσότερο μέσα στην οθόνη κάνοντας περισσότερους κύκλους.

Τέλος, τα jets κάνουν γρήγορες απευθείας επιθέσεις προς τον παίκτη, ορμώντας με μεγαλύτερες καμπύλες στην κυκλική τους κίνηση.

#### Μεσαία εχθρικά αεροσκάφη

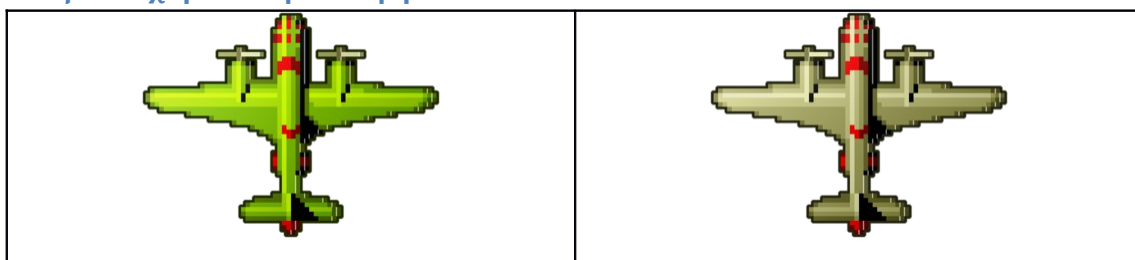


Εμφανίζονται λιγότερο συχνά σε σχέση με τα μικρά αεροσκάφη και θέλουν περισσότερα από ένα χτυπήματα για να καταστραφούν. Μπορεί να εμφανιστούν με ένα από τα 3 μοτίβα:

- Εμφανίζονται από το πάνω μέρος της οθόνης κάνουν ένα κύκλο και συνεχίζουν προς τα κάτω.
- Εμφανίζονται από το κάτω μέρος της οθόνης και ανεβαίνουν προς τα πάνω
- Εμφανίζονται σε σμήνος (μπορεί να εμφανιστούν και από τις δυο πλευρές)

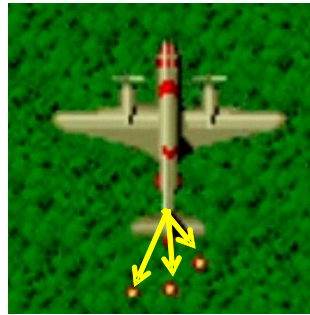
Τα γκρι αεροσκάφη αντέχουν ακόμα περισσότερα χτυπήματα και έχουν μεγαλύτερη ταχύτητα σε σχέση με τα πράσινα.

#### Μεγάλα εχθρικά αεροσκάφη



Είναι οι μεγαλύτεροι εχθροί που συναντάει ο παίκτης εντός πίστας και έχουν καλύτερη άμυνα από τα μικρά και μεσαία αεροσκάφη. Πετούν πάντα από το κάτω μέρος της οθόνης προς τα πάνω, σταματάνε κάπου ψηλά και αρχίζουν να κινούνται προς τα πλαϊνά ρίχνοντας παράλληλα πολλές σφαίρες (πάνω από 3 που απλώνονται σα βεντάλια – βλ. εικόνα παρακάτω) στον παίκτη. Ο παίκτης πρέπει να τα χτυπήσει πολλές φορές για να τα καταστρέψει. Όπως συμβαίνει και με τα μεσαία αεροπλάνα, τα γκριζα πετούν πιο γρήγορα και χρειάζονται ακόμα περισσότερες σφαίρες για να καταρριφθούν. Επίσης τα γκρι ρίχνουν

περισσότερες σφαίρες και σε πιο πολλές κατευθύνσεις από ότι τα πράσινα. Ένα παράδειγμα μεγάλου αεροσκάφους που πυροβολεί είναι στην παρακάτω εικόνα.





## Power-ups

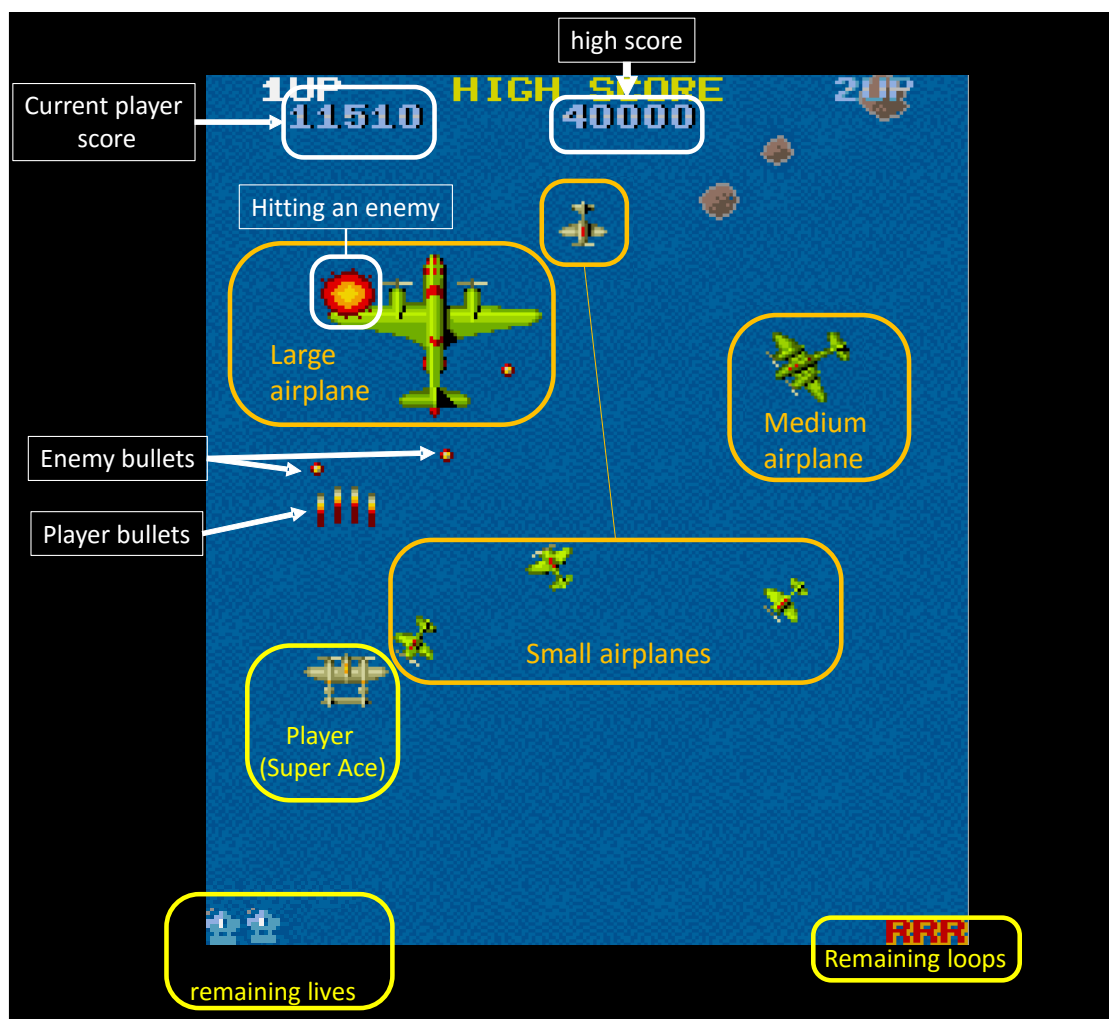


Για τα power-ups εμφανίζεται ένα σμήνος από κόκκινα αεροπλανάκια. Αν ο παίκτης καταφέρει να τα καταστρέψει όλα, τότε το τελευταίο θα αποκαλύψει ένα εικονίδιο POW που αντιστοιχεί σε κάποιο μπόνους. Στο original παιχνίδι εμφανίζεται ένα power-up σε κάθε πίστα, όμως στη δικιά σας εκδοχή θα πρέπει να εμφανιστούν όλα σε αυτή τη 1 πίστα που θα φτιάξετε.

Εικονίδιο	Όνομα	Περιγραφή
	<b>Quad Gun</b>	Το Super-Ace μπορεί να πετάει 4 σφαίρες μαζί τη φορά.
	<b>Enemy Crash</b>	Καταστρέφει μονομιάς όλα τα εχθρικά αεροσκάφη που βρίσκονται στην οθόνη την εκάστοτε στιγμή.
	<b>Side Fighters</b>	2 φιλικά αεροσκάφη έρχονται και «κολλάνε» στο πλάι του super-ace τα οποία i) κινούνται μαζί με τον παίκτη και ρίχνουν από 1 σφαίρα το καθένα κάθε φορά που ο παίκτης πυροβολεί, ii) παραμένουν ενεργά μέχρι να τα καταστραφούν από σύγκρουση με άλλο αεροσκάφος ή σφαίρα ή μέχρι να χάσει ζωή ο παίκτης και τέλος iii) αν ο παίκτης είχε ήδη κάποιο άλλο μπόνους, τότε παίρνει τα αεροσκάφη και επιπλέον πόντους.
	<b>Extra Life</b>	Δίνει μια έξτρα ζωή στον παίκτη.
	<b>No Enemy Bullets</b>	Τα μικρά και τα μεσαία εχθρικά αεροσκάφη δεν μπορούν να πυροβολήσουν για ένα μικρό χρονικό διάστημα.

	<b>Extra Loop</b>	Ο παίκτης μπορεί να κάνει έναν επιπλέον ελιγμό (loop) για να αποφύγει εχθρικά πυρά. Αν δε χρησιμοποιηθεί μέχρι το τέλος της πίστας, δίνει εξτρά πόντους στον παίκτη.
	<b>1000 Points</b>	Δίνει 1000 πόντους.

## Η οθόνη του παιχνιδιού



## Αρχή και τέλος πίστας

Στην αρχή της πίστας το αεροπλάνο του παίκτη (Super Ace) βρίσκεται πάνω σε ένα αεροπλανοφόρο. Η οθόνη στην αρχή του παιχνιδιού είναι όπως φαίνεται παρακάτω στα αριστερά (στη δικιά σας υλοποίηση το “Last 32 stage” μπορεί να αλλάξει ή να λείπει εντελώς ανάλογα με το πόσες πίστες θα φτιάξετε).



Η πίστα τελειώνει όταν φτάσουμε στο επόμενο αεροπλανοφόρο (βλέπε εικόνα πάνω δεξιά). Σε εκείνο το σημείο σταματάνε να εμφανίζονται εχθρικά αεροσκάφη, τυχών εναπομείναντα αεροσκάφη δεν αλληλοεπιδρούν με τον παίκτη και εμφανίζονται στατιστικά του παιχνιδιού σχετικά με τους εχθρούς που καταρρίφθηκαν και τους πόντους που κέρδισε ο παίκτης.

## Ενότητα 2: Κατασκευαστικές οδηγίες

Οι οδηγίες που ακολουθούν ορίζουν τη συμπεριφορά του παιχνιδιού και **δανείζονται** από το αυθεντικό παιχνίδι. Παρόλο που η συμπεριφορά του δικού σας παιχνιδιού **πρέπει** να είναι αυτή που περιγράφεται παρακάτω, η υλοποίησή της δεν χρειάζεται να είναι αυτή που προτείνουμε. Μπορείτε να υλοποιήσετε την ίδια συμπεριφορά με το δικό σας τρόπο αν το επιθυμείτε. Αρκετές από τις συμπεριφορές που θα περιγραφούν παρακάτω **πρέπει** να είναι και [παραμετροποιήσιμες](#)).

### Έλεγχος χρονισμού της δράσης

Για τον έλεγχο της δράσης απαιτείται προσεκτικός χρονισμός σε αρκετά σημεία του παιχνιδιού. Αυτό μπορεί να υλοποιηθεί εύκολα με ένα ειδικό τύπο animation και animator, τα λεγόμενα **time ticks** (TimeTickAnimation, TimeTickAnimator). Η υλοποίηση τους είναι πολύ απλή και βοηθούν στη χρονοδρομολόγηση, στο ίδιο πάντα thread, διαφόρων ενεργειών. Ο ορισμός τους σκιαγραφείται παρακάτω.

```
class TickAnimation : public Animation {
public:
    using OnTick = std::function<void(void)>;
private:
    delay_t      delay;
    byte         repetitions; // 0 => forever
    OnTick       onTick;

public:
    template <typename Tfunc>
    void SetOnTick (const Tfunc& f)
    { onTick = f; }
    void SetForever (void)
    { repetitions = 0; }
```



```

    TickAnimation (animid_t id) :
        Animation      (id),
        Delay           (0),
        Repetitions     (1),
        onTick          (nullptr)
    {}
};

class TimerTickAnimator : public Animator {
public:
    void Progress (timestamp_t currTime);
    TimerTickAnimator (TickAnimation* tick);
};

```

Κάθε φορά που συμπληρώνεται ένα delay, εκτελείται το tick action, ενώ η διαδικασία αυτή λέγεται ένα time tick. Όταν συμπληρωθούν τόσα time ticks όσα η τιμή του repetitions, ο animator σταματάει, ενώ εάν repetitions == 0, το animation δεν σταματάει ποτέ (εκτός και εάν γίνει κλήση του Stop function).

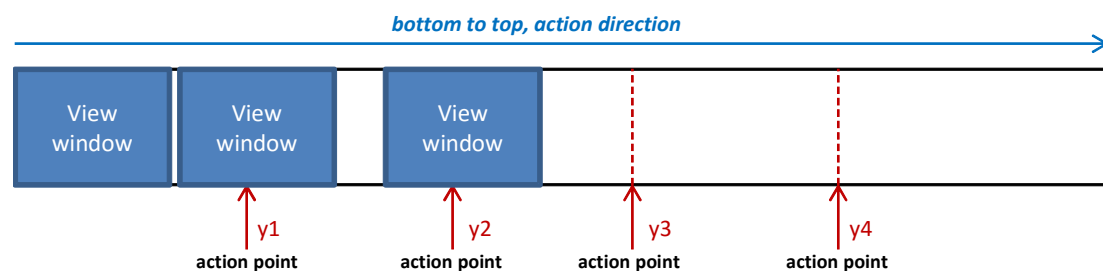
## Κατασκευή της πίστας

Στο κανονικό παιχνίδι υπάρχει tilemap από το οποίο συντίθεται το terrain της πίστας. Εφόσον όμως δεν υπάρχει κάποια αλληλεπίδραση του terrain με το χαρακτήρα μας, και εφόσον δε θέλουμε να φτιάξουμε πολλές πίστες, στη δικιά μας έκδοση μπορούμε να εφαρμόσουμε οποιαδήποτε από τις παρακάτω τεχνικές. Ένα ενδεικτικό background image εσωκλείεται στο zip file του project.

### α) Χρήση ενός large background bitmap

Φυσιολογικά αυτή η τεχνική δε συνίσταται, αλλά στο συγκεκριμένο παιχνίδι με το terrain να είναι μία «λωρίδα» η υλοποίηση της είναι εφικτή. Κάθε στιγμή θα είναι ορατό μόνο ένα τμήμα από το background image, αυτό που περιέχεται στο view window, ενώ κατά τη διάρκεια του παιχνιδιού θα αλλάζει το view window με κάποιο scrolling animation.

Αρχικά το view window είναι στο bottom part του bitmap (full width). Τα scrolling animations θα έχουν  $dx=0$ ,  $dy < 0$  (upwards), repetitions όσα χρειάζονται, και delay «όσο πρέπει» ώστε σε συνδυασμό με το dy να προκύπτει smooth scrolling.



Στην εικόνα φαίνεται η τοπολογία του bitmap, καθώς και η χρήση action points (οριζόντιες γραμμές) – όταν το view window κάνει cross ένα action point τότε πυροδοτείται η ενέργεια που σχετίζεται με αυτό και γίνεται removed.

### β) Χρήση πολλών screen-height bitmaps για τη συρραφή του terrain

Ουσιαστικά είναι ένα collage και η μόνη διαφορά είναι ο έλεγχος που απαιτείται είναι να προσδιορίσετε τα δύο τμήματα των bitmaps που πρέπει να ζωγραφιστούν.



### γ) Διαχωρισμός του background από τα terrain objects

Για το βασικό background (θάλασσα), χωρίς τα νησιά και τα πλοία – αεροπλανοφόρο, μπορούμε να έχουμε μία από τις δύο προηγούμενες τεχνικές εάν επιθυμούμε οπτικές διαφοροποιήσεις. Επειδή όμως έχουμε απλώς ένα fixed bluish background, φτιάχνουμε ένα terrain χωρίς grid (δε χρειάζονται έλεγχοι για την κίνηση του χαρακτήρα), με κατάλληλη τοποθέτηση στο initialization του παιχνιδιού των world sprites (islands, ships). Το terrain data structure είναι ένα singleton class που απλώς περιέχει τα εξής:

- *terrainWidth, terrainHeight*
- *viewWinHeight* (το width είναι πάντα ίδιο με το terrain width)
- *std::list<Sprite\*> terrainObjects*, τα οποία δημιουργούνται στην αρχή και τοποθετούνται στις θέσεις που πρέπει (οι θέσεις αυτές υπολογίζονται «με το μάτι» πάνω στο terrain map που σας δίνεται)
- *startYPos* του view window (αρχικά *terrainHeight – viewWinHeight*)
- μία μέθοδο *GetVisibleObjects()* που επιστρέφει τη λίστα με τα sprites που είναι visible (overlap with) μέσα στο view window (να είναι cached αυτή η λίστα, να υπολογίζεται μόνο κάθε φορά που αλλάζει το view window)
  - ο επειδή τα world sprites είναι λίγα δε χρειάζεται να υλοποιήσουμε την τεχνική με split του terrain σε sectors (δηλ. ένα mega grid) όπου κάθε sector θα είχε τη λίστα των sprites που είναι μέσα ή επικαλύπτονται με αυτό.

### Κίνηση εχθρικών αεροσκαφών

Όλες οι κινήσεις των εχθρικών αεροσκαφών θα πρέπει να εκτελούνται με κατάλληλα animations. Συγκεκριμένα, κάποιες απλές κινήσεις όπως η κατακόρυφη κίνηση μικρών αεροσκαφών μπορούν να υλοποιηθούν με moving animations, κάποιες άλλες που εμπεριέχουν απλά αλλαγή του frame του αεροπλάνου μπορούν με υλοποιηθούν με frame list animations, ενώ οι περισσότερες περιπτώσεις που χρειάζονται παράλληλα και κίνηση και αλλαγή frame απαιτούν moving path animations.

Για τις περιστροφές των αεροσκαφών δε θα χρειαστείτε rotation (θα μπορούσε πάντως να υλοποιηθεί και έτσι) αλλά θα φροντίσετε τα animation films των εχθρών να έχουν διαφορετικά frames για τις διάφορες κλίσεις του διαστημοπλοίου (αντίστοιχα με το sprite-sheet που σας δίνεται στις αναφορές) ώστε η περιστροφή να γίνεται με κατάλληλο συνδυασμό μετατόπισης και αλλαγής frame.

Για τις κινήσεις που κατευθύνονται προς το μέρος του παίκτη θα χρειαστείτε να δημιουργήσετε δυναμικά animations λαμβάνοντας υπόψη την τρέχουσα θέση του παίκτη τη χρονική στιγμή που δρομολογείται το animation.

Τέλος, οι περισσότερες κινήσεις των εχθρικών αεροσκαφών αποτελούνται από συνδυασμό κινήσεων και οι οποίες υλοποιούνται με διαδοχικά animations. Συγκεκριμένα, μπορείτε όταν ξεκινάτε τον κάθε animator να βάζετε στην OnFinish callback μια συνάρτηση που θα

λάβει υπόψη τα δεδομένα που θα ισχύουν στο τέλος του πρώτου animation ώστε να προγραμματίζει κατάλληλα το επόμενο animation. Επίσης είναι πιθανό να χρειαστείτε να υπολογίσετε διάφορα πράγματα μετά από κάθε στάδιο του animation progress. Για να το κάνετε αυτό, μπορείτε να προσθέσετε στο animator class μια OnProgress callback που θα μπορεί να θέσει ο προγραμματιστής ώστε να καλείται μετά από κάθε progress step. Η αλλαγή αυτή σκιαγραφείται στον κώδικα που φαίνεται παρακάτω.

```
class Animator {
public:
    typedef void (*ProgressCallback)(Animator*, void*);
    εναλλακτικά:
    using OnProgress = std::function<void(Animator*)>;
private:
    ProgressCallback onProgress;
    void* progressClosure;
    εναλλακτικά:
    OnProgress onProgress;
protected:
    void NotifyProgressed (void)
        { if(onProgress)(*onProgress)(this, progressClosure); }

public:
    ProgressCallback GetProgressCallback(void) const
        { return onProgress; }
    void* GetProgressClosure (void) const
        { return progressClosure; }
    void SetOnProgress (ProgressCallback f, void* c = 0)
        { onProgress = f, progressClosure = c; }
    εναλλακτικά:
    template <typename Tfunc>
    void SetOnProgress (const Tfunc& f) { onProgress = f; }
    //Rest of the Animator class as before
};

//Also update subclasses to call NotifyProgressed in Progress
class MovingAnimator : public Animator {
public:
    void Progress (timestamp_t currTime) {
        while (...) {
            //original code for progressing
            NotifyProgressed();
        }
        // Rest of the MovingAnimator class as before
    }
};
```

## Side fighters

Τα φιλικά αεροσκάφη που έρχονται και «κολλάνε» στο πλάι του Super Ace όταν ο παίκτης πάρει το αντίστοιχο power-up μπορούν να υλοποιηθούν ως ξεχωριστά sprites τα οποία περιορίζονται από την κίνηση του Super Ace (κινούνται όπως κινείται και αυτό) αλλά και περιορίζουν την κίνησή του (ένα πλαϊνό αεροσκάφος δε μπορεί να βγει από τα όρια της οθόνης, οπότε το Super Ace δε μπορεί να πάει τέρμα δεξιά ή αριστερά έχοντας τα πλαϊνά αεροσκάφη). Συγκεκριμένα, όταν ο παίκτης πάρει το power-up, τα δύο φιλικά αεροσκάφη

δημιουργούνται στο πάνω μέρος της οθόνης και κάνουν ένα animation προς το μέρος του Super Ace. Όταν φτάσουν σε αυτό σταθεροποιούνται, το ένα στα δεξιά του και το άλλο στα αριστερά του, και η κίνησή τους καθορίζεται πλέον από την κίνηση του Super Ace. Το ίδιο ισχύει για τον τρόπο που πυροβολούν. Κατά τα άλλα, είναι ξεχωριστά sprites, τα οποία κάνουν collide με τους εχθρούς και τις σφαίρες τους. Το μόνο ακόμα που χρειάζεται είναι να ξέρουμε αν χτυπηθούν από κάποιον εχθρό ώστε να ενημερωθεί κατάλληλα ο Super Ace. Η λειτουργικότητα κίνησης και χειρισμού των φιλικών αεροσκαφών σκιαγραφείται στον παρακάτω κώδικα. Επεκτείνουμε το Sprite class να υποστηρίζει attached named sprites τα οποία ακολουθούν αυτομάτως (άρα είναι constrained) την ίδια κίνηση με το πατρικό (parent) sprite.

```
class Sprite : public LatelyDestroyable {
using Attached = std::map<std::string, Sprite*>;

Attached      attached;
Sprite*       parent;
std::string   name;

    void MoveAttached (int dx, int dy) {
        for (auto& i : attached)
            i.second->Move(dx,dy);
    }
public:
    virtual void FilterMotion (int* dx, int* dy) const
        { /* default is unfiltered motion */ }

    void Attach (Sprite* s, const std::string& name)
        { attached[name] = s; s->parent = this; s->name = name; }

    void Detach (const std::string& name, bool destroy) {
        auto i (attached.find(name));
        assert(i != attached.end());
        auto* s (i->second);
        s->parent = nullptr;
        s->name.clear();
        attached.erase(i);
        if (destroy)
            s->Destroy();
    }

    Sprite* GetAttached (const std::string& name) {
        auto i (attached.find(name));
        return i != attached.end() ? i->second : nullptr;
    }

    virtual void Move (int dx, int dy) {
        FilterMotion(&dx, &dy);
        <η βασική λογική της Move εδώ>;
        MoveAttached(dx, dy);
    }

    virtual void Destroy (void) { // supports auto detach policy
        LatelyDestroyable::Destroy();
        if (parent) // is attached
            parent->Detach(name, false);
    }
};
```

```
    }  
    // Rest of the Sprite class as before  
};
```

```

#define LEFT_FIGHTER "left.fighter"
#define RIGHT_FIGHTER "right.fighter"

class SideFighter : public Sprite {
public:
    SideFighter(Sprite* ace, const std::string& name)
        { ace->Attach(this, name); }
    void Fire (void) { /* spawn bullet */ }
};

class SuperAce : public Sprite {
public:
    virtual void FilterMotion (int* dx, int* dy) const override {

        int old_x = x;
        int old_y = y;
        int new_x = x + *dx;
        int new_y = y + *dy;

        //adjust new_y if out of screen bounds
        int max_y = Screen.GetHeight() - this->GetHeight();
        if (new_y < 0)
            new_y = 0;    //min_y == 0
        else
            if (new_y > max_y)
                new_y = max_y;

        //adjust new_x so that if there are any fighters
        // present they do not go beyond screen bounds
        auto* left = GetAttached(LEFT_FIGHTER);
        auto* right = GetAttached(RIGHT_FIGHTER);

        int min_x = left ? left->GetWidth() : 0;
        int max_x = Screen.GetWidth() - this->GetWidth();
        if (left)
            max_x -= left->GetWidth();

        if (new_x < min_x)
            new_x = min_x;
        else
            if (new_x > max_x)
                new_x = max_x;

        // now filter motion offset with actual motion possible

        *dx = new_x - old_x;
        *dy = new_y - old_y;
    }

    void Fire (void) {
        /* spawn SuperAce bullets */
        if (auto* left = GetAttached(LEFT_FIGHTER))
            static_cast<SideFighter*>(left)->Fire();
        if (auto* right = GetAttached(RIGHT_FIGHTER))
            static_cast<SideFighter*>(right)->Fire();
    }
    ...
};

```

## Προσαρμογές (configuration facilities)

Μπορείτε να έχετε έλεγχο ορισμένων χαρακτηριστικών του παιχνιδιού μέσω configuration files. Ορισμένα από τα επιθυμητά configurations μέσω των οποίων θα έχετε τη δυνατότητα να κάνετε το παιχνίδι σας να μοιάζει κάθε φορά και διαφορετικό είναι τα εξής:

- Αρχικός αριθμός ζώων
- Αρχικός αριθμός loops
- Ταχύτητα κίνησης και αριθμός των αντιπάλων
- Ταχύτητα κίνησης του παίκτη
- Ρυθμός και ταχύτητα σφαιρών
- Συχνότητα εμφάνισης και αριθμός των κόκκινων αεροσκαφών που δίνουν τα power-ups

## Pause – Resume

Στο παιχνίδι σας είναι **υποχρεωτικό** να υλοποιηθεί ο μηχανισμός του pause και του resume. Κάθε φορά που ο παίχτης επιλέγει να «παγώσει» το παιχνίδι, θα πρέπει να εμφανίζεται μια οθόνη που θα ενημερώνει των χρήστη ότι το παιχνίδι έχει σταματήσει προσωρινά.

---

## Ομαδική εργασία και παράδοση

---

Η εργασία μπορεί να περατωθεί από ομάδα τριών (3) το πολύ ατόμων. Το ποσοστό της τελικής βαθμολογίας που καταλαμβάνει η εργασία σας είναι 45% και σε εξαιρετικές περιπτώσεις πολύ καλών αποτελεσμάτων μπορεί να γίνει και 50%. Η ημερομηνία παράδοσης/εξέτασης του project θα είναι την περίοδο που μεσολαβεί μετά την λήξη της εξεταστικής και την έναρξης του νέου εξαμήνου.

Η παράδοση / εξέταση της όπως συνηθίζεται θα γίνει παρουσία όλων από τον διδάσκοντα (μπορείτε να φέρετε και invited άτομα αρκεί να έχουν δηλωθεί πιο νωρίς ώστε αν χρειαστεί να προγραμματίσουμε άλλο χώρο εξέτασης).

Φροντίστε η αρχική οθόνη του παιχνιδιού να περιέχει: τα ονόματα της ομάδας σας με λατινικούς χαρακτήρες και κεφάλαια γράμματα καθώς και την ένδειξη “University of Crete \n Department of Computer Science \n CS-454. Development of Intelligent Interfaces and Games \n Term Project, Fall Semester 2015”.

---

## Αναφορές

---

1. [http://strategywiki.org/wiki/1942/How\\_to\\_play](http://strategywiki.org/wiki/1942/How_to_play)
2. <http://www.sprites-resource.com/arcade/1942/sheet/61738/> Original Sprite Sheet



3. [https://johnsontechnology.wikispaces.com/file/view/1945\\_sprites.bmp/362699502/1945\\_sprites.bmp](https://johnsontechnology.wikispaces.com/file/view/1945_sprites.bmp/362699502/1945_sprites.bmp) Alternative Sprite Sheet
4. [http://members.iinet.net.au/~tmorrow/emulation/map\\_1942/map\\_1942\\_map.html#StartOfMap](http://members.iinet.net.au/~tmorrow/emulation/map_1942/map_1942_map.html#StartOfMap) game map (all stages)