

Θέμα 1^ο 25 Μονάδες

(α) 5 Μονάδες. Ποιες είναι οι βασικές κατηγορίες λειτουργικότητας των εφυών διεπαφών και ποια είναι τα χαρακτηριστικά της κάθε κατηγορίας;

1. Χειροκίνητη προσαρμογή από τον χρήστη του συστήματος - **manual configuration**

- Στην περίπτωση αυτή προσφέρονται διάφορες δυνατότητες στον τελικό χρήστη να προσαρμόσει τα χαρακτηριστικά της διεπαφής στις δικές του εκάστοτε προτιμήσεις και ανάγκες.
- Το είδος των προσφερόμενων λειτουργιών προσαρμογής, καθώς και ο τρόπος υλοποίησης τους, ποικίλει αρκετά.

2. Αυτόματη προσαρμογή κατά τη χρήση με παρακολούθηση - **adaptation during use**

- Στην περίπτωση αυτή η διεπαφή παρακολουθεί διαρκώς την αλληλεπίδραση με τον χρήστη (user monitoring),
- καθώς και διάφορες παραμέτρους / χαρακτηριστικά του περιβάλλοντος χρήσης (environment parameters), αναπροσαρμόζοντας δυναμικά κάποια στοιχεία της για την επίτευξη της καλύτερης δυνατής ποιότητας αλληλεπίδρασης.

3. Αυτόματη προσαρμογή πριν την έναρξη λειτουργίας - **adaptation before use**

- Η διεπαφή έχει πληροφορία για βασικές παραμέτρους / χαρακτηριστικά του χρήστη
- Πριν την εκτέλεση προσαρμόζει / επιλέγει διάφορα στοιχεία της ώστε να ανταποκρίνονται με βέλτιστο τρόπο στον ήδη γνωστό τελικό χρήστη

(β) 5 Μονάδες . Ποια από τα παρακάτω είναι τα χαρακτηριστικά των αρχείων προσαρμογής και ποια δεν είναι.

1. Το συντακτικό τους περιγράφεται από κανονική γλώσσα (regular languages).

Ναι [X] Όχι []

2. Απαιτούν την προσεκτική επιλογή των προσαρμοσμών χαρακτηριστικών της διεπαφής.

Ναι [X] Όχι []

3. Η γλώσσα προσαρμογής έχει συνήθως απλό συντακτικό και λίγες λέξεις κλειδιά.

Ναι [X] Όχι []

4. Απαιτείται η κατασκευή parser που διαβάζει της ... και παράγει κώδικα που θα εφαρμόζει τις προσαρμογές πάνω στο κυρίως πρόγραμμα

Ναι [X] Όχι []

5. Μπορούν να γίνονται αρκετά πολύπλοκα προσφέροντας μεγάλες δυνατότητες προσαρμογών.

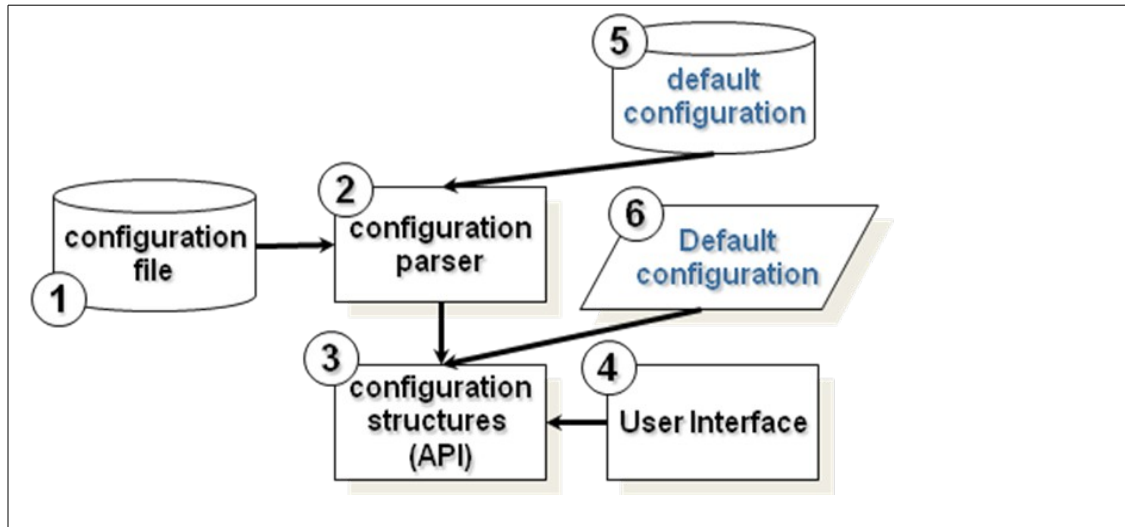
Ναι [X] Όχι []

6. Είναι πάντοτε διαθέσιμα με ένα αντίστοιχο ...βάσης στις παραμέτρους προσαρμογής και τις τιμές που έχουν λάβει

Ναι [] Όχι [X]

(γ) 5 Μονάδες. Σημιώστε τι περιγραφές που λείπουν στην διπλανή αρχιτεκτονική(1,2,3), και δώστε με μία πρόταση παρακάτω τι ρόλο έχουν τα τμήματα 1, 2, 3, που συμπληρώνετε.

■ **Τεχνικές υποστήριξης χειροκίνητης προσαρμογής**



1) Αρχεία Προσαρμογής – Configuration files

- Παρέχονται αρχεία κειμένου για τον ορισμό των προσαρμογών.
Το εξωτερικό αρχείο προσαρμογής.

2) Ο configuration parser διαβάζει τα configuration files.

3) Οι εσωτερικές δομές των configuration data (μία ή περισσότερες κλάσεις).

1. Το εξωτερικό αρχείο προσαρμογής.

2. Ο parser που διαβάζει τα configuration files.

3. Οι εσωτερικές δομές των configuration data (μία ή περισσότερες κλάσεις).

4. Η υλοποίηση της διεπαφής (έχει τη δική της κατάτμηση).

5. Το default configuration ως εξωτερικό αρχείο.

6. Το default configuration ως εσωτερικά δεδομένα ή εντολές αρχικοποίησης.

(δ) 5 Μονάδες. Ποια από τα παρακάτω είναι / δεν είναι πλεονεκτήματα των αρχείων προσαρμογής.

(Διάλεξη 1^η)

1. Είναι πολύ εύκολη η υλοποίησή τους σε κώδικα,
Ναι [] Όχι [X]

2. Διευκολύνουν την κλιμάκωση (scalability) με εισαγωγή στη «γλώσσα προσαρμογής» νέων χαρακτηριστικών της διεπαφής που είναι προσαρμόσιμα.
Ναι [X] Όχι []

3. Οι εντολές αρχικοποίησης στον κώδικα της διεπαφής ενοποιούνται σε μία κλάση προσαρμογής.
Ναι [X] Όχι []

4. Αποφεύγονται οι εντολές αρχικοποίησης με hard-coded values στον κώδικα.
Ναι [X] Όχι []

5. Υπάρχει ομοιογένεια των διεπαφών σε διαφορετικά installations της εφαρμογής καθώς υπάρχει ένα αμετάβλητο configuration file.
Ναι [X] Όχι []

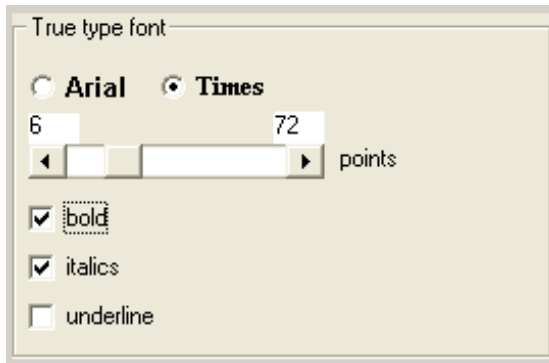
6. Μπορεί να γίνει ενεργοποίηση κάποιας άλλης προσαρμογής ενώ το σύστημα τρέχει (change active configuration on-the-fly).
Ναι [X] Όχι []

7. Μπορούν να χρησιμοποιηθούν στην κατασκευή τους έτοιμα εργαλεία όπως lexical analyzer generators και parser generators.
Ναι [X] Όχι []

(ε) 5 Μονάδες. Σας δίνονται τα παρακάτω micro interfaces για προσαρμογή χαρακτηριστικών της διεπαφής. Κάθε ένα αντιστοιχεί σε μία ειδική εσωτερική κλάση «τύπου προσαρμόσιμων δεδομένων». Θυμίζουμε ότι όλοι αυτοί οι τύποι κληρονομούν από τον τύπο Property, ενώ οι σύνθετοι τύποι κληρονομούν από τον τύπο AggregateProperty. Γράψτε στον πίνακα που ακολουθεί τον τύπο που αντιστοιχεί σε κάθε περίπτωση και γιατί.

1

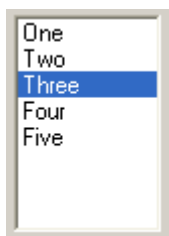
2



3

4

5



6

- 1: EnumaratedProperty
- 2: IntRangeProperty
- 3: BooleanProperty
- 4: NumericProperty
- 5: FontProperty
- 6: EnumaratedProperty

(Διάλεξη 2)

Θέμα 2^ο 25 Μονάδες

(α) 5 Μονάδες. Συμπληρώστε τα κενά στην αρχιτεκτονική προσαρμογής «πριν τη χρήση» στο διπλανό σχήμα περιγράφοντας σύντομα το ρόλο κάθε τμήματος στον παρακάτω πίνακα και τοποθετώντας τις κατάλληλες αρχιτεκτονικές συνδέσεις πάνω στο ίδιο το σχήμα.

Context information server (CIS)

Context
Information

Server

- ♦ Να παρέχει τις τιμές των χαρακτηριστικών του χώρου χρήσης - *context attribute values*, τόσο για την πλατφόρμα όσο και για το περιβάλλον.
- ♦ Στατικά χαρακτηριστικά – *static attributes*, δηλ. αυτά που παραμένουν αμετάβλητα κατά τη διάρκεια της αλληλεπίδρασης (π.χ. περιφερειακές συσκευές αλληλεπίδρασης)
- ♦ Δυναμικά χαρακτηριστικά – *dynamic attributes*, δηλ. αυτά που μπορούν να αλλάζουν τιμές κατά την αλληλεπίδραση (π.χ. εξωτερικός θόρυβος)


User information server (UIS)

- ♦ Να παρέχει πληροφορία για το προφίλ του χρήστη, είτε:
 - Κατά την έναρξη της αλληλεπίδρασης, δηλ. ως προϋπάρχουσα γνώση
 - Δυναμικά, με αναγνώριση χαρακτηριστικών μέσω παρακολούθησης της αλληλεπίδρασης
 - ώστε να καλύπτεται και η περίπτωση προσαρμογής κατά τη χρήση.

Decision making component (DMC)

- ♦ Ουσιαστικά αποφασίζει, δεδομένου του προφίλ του χρήστη και του χώρου χρήσης, ποια είναι τα κατάλληλα τμήματα της διεπαφής που πρέπει να «συμμετέχουν» τελικά στην συνολική διεπαφή.
- ♦ Πρόκειται για έναν πυρήνα που περιλαμβάνει υλοποιημένου κανόνες

Μονάδες. Σκιαγραφήστε παρακάτω τη δομή του game loop σε ψευδοκώδικα, χρησιμοποιώντας μία πρόταση για κάθε συνάρτηση που φαίνεται να καλείται μέσα σε αυτό.

<p>Επεξηγήσεις</p> 	<pre>Void GameMainLoop(){ While (NotFinished()) { Input Management (); Artificial Intelligence (); Animation Progress (); Collision Checking (); Rendering (); FPS Calculation (); System Loop Dispatching (); } }</pre>
--	--

(γ) 10 Μονάδες. Ένας συνάδελφος σας ρωτάει το εξής: «Γνωρίζω ότι στην κατασκευή παιχνιδιών στοχεύετε στην επίτευξη όσο το δυνατόν μεγαλύτερου frame rate, π.χ. μια έκδοση στο 240 ότι δεν είναι παράλογο; Αφού ποτέ ένας άνθρωπος δεν πρόκειται να διακρίνει την αλληλουχία τους με αυτή την ταχύτητα ενώ επιπλέον λόγω του χαμηλού frequency που έχουν οι οθόνες σήμερα τίθεται και ένα ζήτημα του hardware ως προς τα πόσα frames τελικά μπορεί κανείς να παρακολουθήσει στην οθόνη. Γιατί λοιπόν όλη αυτή η φασαρία;» Τι έχετε να του απαντήσετε ως επίδοξοι game developers;

Δεν στοχεύουμε στην υλοποίηση παιχνιδιών όπως οι μεγάλες βιομηχανίες και για το λόγο αυτό είναι καλό να υλοποιούμε έξυπνα παιχνίδια με αρκετό AI.

Δεν χρειάζεται να περάσεις τα 200 frames εφόσον δεν μπορεί να τα δει το μάτι, και μπορούμε να τα χρησιμοποιήσουμε

Οι οθόνες έχουν μεγαλύτερο ρυθμό ανανέωσης που φτάνει στο 100 HZ, μέσα σε αυτές τις ανανέωσης. Για να ξεπεράσεις το 200 frames πρέπει να κάνεις δύο κύκλους μέσα σε ένα sec και δεν έχει κανένα νόημα εφόσον δεν θα υπάρχει διαφορά στην εικόνα, και μπορεί να χρησιμοποιηθεί ο δεύτερος κύκλος για το AI για παράδειγμα.

εάν πετύχουμε πολύ μεγάλο frame rate, π.χ. 150, ζωγραφίζουμε χωρίς λόγο καθώς και με 100, δηλ αν παραλείψουμε 50 rendering calls, πετυχαίνουμε καλύτερο αποτέλεσμα και μπορούμε να χρησιμοποιήσουμε το χρόνο που μένει για το AI

- Εάν το game loop μπορεί πάντοτε να τρέχει με ταχύτητα μεγαλύτερη του frequency ή όταν είναι πιο αργό αυτό είναι ανεκτό

- ♦ Δεν μας απασχολεί καμία χρονοδρομολόγηση και απλά κάνουμε *Vsync* χωρίς να μας ενδιαφέρει το γεγονός ότι σπαταλάμε χρόνο, αφού ούτως ή άλλως δεν τον χρειαζόμαστε
- Εάν το *game loop* είναι αργότερο της συχνότητας της οθόνης και αυτό έχει ορατά αρνητικά αποτελέσματα τότε
 - ♦ Βελτιστοποιούμε το AI ή / και το *rendering*
 - ♦ Εάν δεν βελτιστοποιούνται παραπάνω και δεν μπορούμε να έχουμε καμία βελτίωση τότε
 - Είτε μειώστε την ποσότητα του AI ή την πολυπλοκότητα των σκηνών ώστε να έχουμε υπολογιστικά πιο light AI ή *rendering*
 - Ή κάνετε *pre-caching* (θα δούμε τέτοιες τεχνικές) όσο παραπάνω μπορείτε ορισμένων *runtime* υπολογισμών μέχρι να έχετε πραγματικά ορατή βελτίωση
- Η τεχνική αυτή είναι ελαφρώς βελτιωμένη λύση του προβλήματος καθυστέρησης λόγω αναμονής της περάτωσης του *display refreshing*
- Πρέπει να υποστηρίζεται από το h/w και δουλεύει ως εξής:
 - ♦ Έχετε την οθόνη, δηλ. το ***frame buffer*** καθώς και ένα ακόμη *video bitmap* ίδιου τύπου που παίζει το ρόλο του ***back buffer***
 - ♦ Το h/w γνωρίζει από πριν τη μνήμη που καταλαμβάνει όχι μόνο ο ***frame buffer*** αλλά και ο ***back buffer***
 - ♦ Όταν αποτυπώσουμε στον *back buffer* τη σκηνή αντί να κάνουμε *VSync* και *blit* στο *frame buffer*, λέμε στο h/w να αλλάξει τους ρόλους των δύο *bitmaps* και να θεωρεί ως *frame buffer* τον *back buffer* και ως *back buffer* τον μέχρι πρότινος *frame buffer*
 - Δηλ. το h/w κάνει *flip* τους ρόλους τους, έτσι γλυτώνουμε το αναγκαστικό *blit* από τον *back-buffer* στο *frame-buffer*
 - Στο επόμενο *refresh* αυτομάτως τα περιεχόμενα του μέχρι πριν *back buffer* θα εμφανιστούν στην οθόνη

(δ) 5 Μονάδες. Ο προηγούμενος φίλος σας τελικά έχει πειστεί και αποφάσισε να εμπλακεί στο πεδίο εφαρμογών των παιχνιδιών. Μετά από μερικά παιχνίδια έχει γίνει κατασκευαστής νέος παιχνιδιού. Σας συναντάει τυχαία στο δρόμο και περιγράφοντας το παιχνίδι του περηφανεύεται ότι «πετυχαίνει σε ορισμένα σημεία της δράσης ακόμα και 240 FPS». Είναι κάτι για το οποίο όντως πρέπει να είναι περήφανος; Ως επαγγελματίες τι ερωτήματα θα του θέτατε σχετικά με αυτό;

Όχι δεν πρέπει να είναι περήφανος καθώς όπως και αναφέραμε και παραπάνω δεν είναι εάν πετύχουμε πολύ μεγάλο *frame rate*, π.χ. 150, ζωγραφίζουμε χωρίς λόγο καθώς και με 100, δηλ αν παραλείψουμε 50 *rendering calls*, πετυχαίνουμε καλύτερο αποτέλεσμα και μπορούμε να χρησιμοποιήσουμε το χρόνο που μένει για το AI.

Θέμα 3^ο 25 Μονάδες

(α) 5 Μονάδες. Εντοπίστε ποιες από τις παρακάτω διατυπώσεις είναι ορθές και ποιες είναι λανθασμένες.

1. Ο γραφικός επιταχυντής χρησιμοποιεί πάντα τμήμα της κεντρικής μνήμης για την αποθήκευση των bitmaps.

Σωστό [] Λάθος [X]

2. Ο frame buffer αποτυπώνεται στην οθόνη με κάθε κλήση αντίστοιχης εντολής του λογισμικού.

Σωστό [] Λάθος [X]

3. Η μνήμη οθόνης (frame buffer) για οποιαδήποτε συγκεκριμένη κάρτα έχει πάντα το ίδιο μέγεθος. (ο frame buffer εξαρτάται από την ανάλυση και τα χρώματα διευκρίνηση)

Σωστό [] Λάθος [X]

4. Οποιαδήποτε δύο bitmaps διαφορετικών διαστάσεων και διαφορετικών bit depths ταυτόχρονα έχουν **πάντοτε** διαφορετικές απαιτήσεις αποθήκευσης στη μνήμη.

Σωστό [] Λάθος [X]

5. Η τομή (ως σύνολο) των χρησιμοποιούμενων χρωμάτων δύο διαφορετικών bitmaps των 8-bits, δηλ. 256 χρωμάτων, εκ των οποίων στο ένα χρησιμοποιούνται 256 διαφορετικά χρώματα, δεν είναι ποτέ κενό σύνολο.

Σωστό [X] Λάθος []

6. Το μαύρο χρώμα σε μία παλέτα 256 χρωμάτων είναι πάντα το index 0 ενώ το λευκό είναι το index 255.

Σωστό [] Λάθος [X]

7. Σε παλέτα 256 χρωμάτων ένα index με τιμή (0, 0, 0) δεν θα εκτυπωθεί απαραίτητα ως μαύρο.

Σωστό [] Λάθος [X]

8. Η βασική λειτουργία blit επιτρέπεται μόνο μεταξύ bitmaps του ίδιου bit depth.

Σωστό [X] Λάθος []

9. Η ταχύτητα του blit από α \rightarrow β για α και β bitmaps εξαρτάται αποκλειστικά και μόνο από το εμβαδόν του blit rectangle.

Σωστό [] Λάθος [X] (εξαρτάται και από το bit depth)

(β) 5 Μονάδες. Αναφέρετε τέσσερις πολύ συνηθισμένες διαστάσεις για tiles . Τι είναι το tile bitmap και τι το tile index ;

Συνήθως χρησιμοποιούμε tiles με διαστάσεις 16 ή 32 pixels, π.χ .

16x16, 16x32, 32x16, και 32x32

Ένα από τα προφανή κατασκευαστικά προβλήματα σε τέτοιου είδους παιχνίδια είναι ότι ο χώρος δράσης, εάν αποθηκευτεί ως bitmap, είναι τρομακτικά μεγάλος (τουλάχιστον για τα σημερινά δεδομένα).

Η τεχνική που χρησιμοποιείται σε αυτού του είδους τα games είναι ο κατακερματισμός του terrain , δηλ. του χώρου του παιχνιδιού σε μικρό αριθμό από επαναλαμβανόμενα και συνδεδεμένα μεταξύ τους bitmaps.

Αυτά τα bitmaps ονομάζονται tiles, δηλ. «πλακίδια», ή «πλακάκια».

Το bitmap που περιέχει τα tiles ενός game το ονομάζουμε tiles bitmap

Η θέση ενός tile μέσα στο tile-bitmap ορίζεται από έναν μοναδικό αριθμό που λέγεται tile index.

(γ) 10 Μονάδες. Στον παρακάτω κώδικα συμπληρώστε το σώμα της τελευταίας συνάρτησης. Ποια χαρακτηριστική δυνατότητα των tile- based games αποτυπώνει ο παρακάτω κώδικας;

```
#define MAX_VIEWS 4
Rect viewWindows[MAX_VIEWS];
Rect displayAreas[MAX_VIEWS];
void Scroll (Rect* vw, HorizScroll, VertScroll);
bool CanScroll (Rect* vw, HorizScroll h);
bool CanScroll (Rect* vw, VertScroll v);
void DisplayTerrainView (Bitmap dest, Rect& vw, Rect& da) {
    for (Dim row = 0; row < vw.h; ++row)
        for (Dim col = 0; col < vw.w; ++col)
            PutTile(
                dest,
                da.x + MUL_TILE_WIDTH(col),
                da.y + MUL_TILE_HEIGHT(row),
                tileBitmap,
                map[row + vw.y][col + vw.x]
            );
}
void DisplayTerrain (void) {
    for (byte i=0; i<MAX_VIEWS; ++i)
        DisplayTerrainView(BackBuffer(), viewWindows[i], displayAreas[i]);
}
```

(δ) 5 Μονάδες. Τι είναι το animation film ; ποιες από τις παρακάτω διατυπώσεις είναι αληθείς και ποιες ψευδείς για τα animation film;

1. Η τοποθέτηση των key frames μέσα στο animation film δεν ακολουθεί κάποια

προκαθορισμένη τοπολογία, αλλά ο σκοπός είναι πάντα η εύκολη πρόσβαση.

Σωστό [X] Λάθος []

2. Το animation film περιέχει πάντα με τη σειρά αλληλουχίας τα frames μιας κίνησης.

Σωστό [] Λάθος [X]

3. Αποφεύγουμε να έχουμε συνήθως πολλές διαφορετικές κινήσεις μέσα στο ίδιο animations film κυρίως για λόγους ευκολίας επεξεργασίας από τους γραφίστες.

Σωστό [X] Λάθος []

4. Αποφεύγουμε να έχουμε πολλές κινήσεις που αφορούν πολλούς χαρακτήρες μέσα στο ίδιο animation film, καθώς αυτό δημιουργεί προβλήματα στην σωστή οργάνωση και ονομασία των αρχείων – films (π.χ. ανά όνομα χαρακτήρα), ενώ εμποδίζει ενδεχομένως τακτικές loading on demand.

Σωστό [X] Λάθος []

5. Για λόγους συμπίεσης φροντίζουμε να αποθηκεύουμε τα animation film σε 8-bits και να τα μετατρέπουμε κατά το loading στο εκάστοτε bit depth της οθόνης.

Σωστό [] Λάθος [X]

6. Δεν συνηθίζεται τα διαδοχικά frames να τοποθετούνται ομοιόμορφα (δηλ. το καθένα να καταλαμβάνει χώρο ενός ορθογώνιου ίδιων διαστάσεων με τα υπόλοιπα), για να αποφεύγεται η κατασπατάληση χώρου αποθήκευσης και μνήμης καθώς οριεμένα μπορεί να απαιτούν μικρότερο bounding box.

Σωστό [] Λάθος [X]

7. Συνιθίζεται να τοποθετούνται τα διάφορα frames με κάποια φυσική αλληλουχία (π.χ. από αριστερά προς τα δεξιά, από πάνω προς τα κάτω)

Σωστό [X] Λάθος []

8. Δεν είναι απαραίτητο τα αυθεντικά films να περιέχουν τα frames σε minimal bounding boxes.

Σωστό [X] Λάθος []

9. Τα οριζόντια films, με ένα animation ανά film, είναι μια καλή και επαρκής τακτική.

Σωστό [X] Λάθος []

10. Ποτέ δεν χρησιμοποιείται το αυθεντικό bitmap ενός film για να κάνουμε κάποιο blite.

Σωστό [] Λάθος [X]

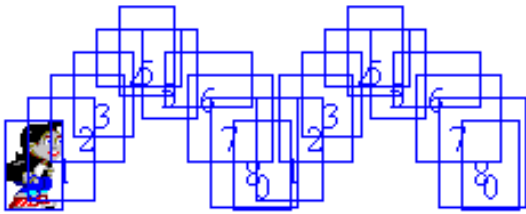
Θέμα 4^ο 25 Μονάδες

(α) 5 Μονάδες. Αναγνωρίστε τα παρακάτω ήδη animations και κατατάξτε τα στην κατάλληλη κατηγορία(σημειώστε τη στον πίνακα που έπεται).

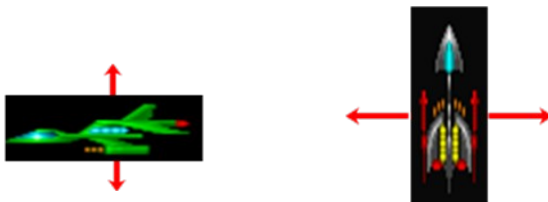
1)



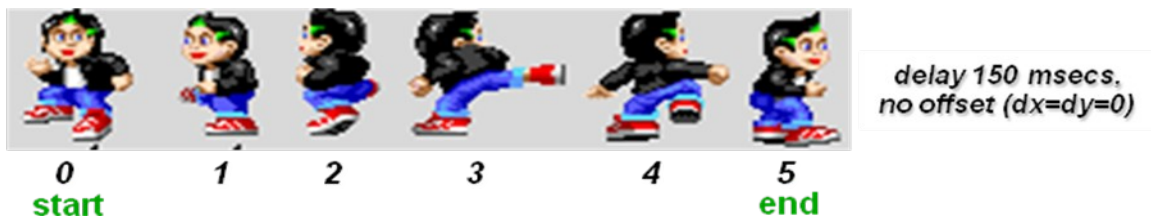
2)



3)



4)



1. Frame list
2. Moving path
3. Moving animation
4. Frame range

Τι είναι ο animator, πόσες κατηγορίες υπάρχουν; πως η ύπαρξη αυτής της κλάσης απλουστεύει τις κλάσεις των animation και sprites;

Ο *animator* γνωρίζει πώς να εφαρμόσει ένα animation instance σε ένα sprite instance (ή σε ένα layer)

- ♦ Οι κατηγορίες που υπάρχουν είναι
 - ***MovingAnimator*** class
 - ***FrameRangeAnimator*** class
 - ***FrameListAnimator*** class
 - ***MovingPathAnimator*** class
 - ***FlashAnimator*** class
 - *scrollingListAnimator*

χρειάζομαι ειδικό τύπο animator για κάθε διαφορετικό τύπο animation

- ενώ είναι απαραίτητος ο διαχωρισμός μεταξύ *sprite animator* και *layer animator*
- προφανώς όταν εφαρμόζεται ένα *animation instance* σε ένα *sprite instance* πρέπει:
 - ♦ το *film instance* του *sprite* να αντιστοιχεί πλήρως σε αυτό το *animation instance*

(β) 5 Μονάδες. Σκιαγραφείστε την animator super class περιγράφοντας σύντομα τις βασικές member functions. Πως διαχειρίζονται μαζικά όλοι animators;

Περιγραφή member functions	Animator Super Class
<pre>void Stop (void); bool HasFinished (void) const { return state != ANIMATOR_RUNNING; } virtual void TimeShift (timestamp_t offset);</pre>	<pre>class Animator { public: typedef void (*FinishCallback)(Animator*,void*); protected: timestamp_t lastTime; animatorstate_t state; FinishCallback onFinish; void* finishClosure;</pre>

<pre>virtual void Progress (timestamp_t currTime)=0; void SetOnFinish (FinishCallback f, void* c=(void*) 0) { onFinish = f, finishClosure = c; } };</pre>	<pre>void NotifyStopped (void); public: void Stop (void); bool HasFinished (void) const { return state != ANIMATOR_RUNNING; } virtual void TimeShift (timestamp_t offset); virtual void Progress (timestamp_t currTime)=0; void SetOnFinish (FinishCallback f, void* c=(void*) 0) { onFinish = f, finishClosure = c; } Animator (void); virtual ~Animator({}); };</pre>
<p>Μαζική διαχείριση των animators</p> <ul style="list-style-type: none"> ● Κάθε instantiated animator είναι registered σε ένα singleton class που, φυσιολογικά, λέγεται animator holder (manager) class ● Αυτό μπορεί εύκολα να υλοποιηθεί στο επίπεδο του super-class καθώς στον constructor μπορεί να γίνεται registered και στον destructor removed ● Εάν όλοι οι animators είναι «μαζεμένοι» σε μία λίστα, μπορώ να κάνω μαζικά σε ένα σημείο progress όλα τα running animations (από τους animators) βάσει του εκάστοτε current time ● Για μεγαλύτερη ταχύτητα μπορώ μάλιστα να έχω δύο λίστες: ● (α) με τους running animators, δηλ. αυτούς που έχουν ήδη γίνει 	

- started,
(β) με τους suspended animators, δηλ αυτούς που έχουν γίνει είτε normally finished, ή explicitly stopped

(γ) 5 Μονάδες. Πως θα υλοποιούνταν η pause και η resume συνάρτηση ώστε να ... συμπεριφέρονται σαν να βρέθηκαν σε «κενά ...» δηλ.να μην επηρεαστεί η αρμονική ρο του animation από την προσωρινή παύση;

Με την *TimeShift* μπορούμε να κάνουμε pause το game εύκολα, αρκεί να κρατήσουμε το χρόνο για τον οποίο έχει γίνει paused και έπειτα να κάνουμε time shift όλους τους animators (δηλ. τα animator instances).

```
void Animator::TimeShift (timestamp_t offset)
{ lastTime += offset; }
```

(δ) 5 Μονάδες. Έστω bitmap A διαστάσεων W = H που γίνεται ... σε bitmap ... Γράψτε μόνο το διπλό for loop (σε ψευτοκώδικα) το οποίο αρχίζει το αλγόριθμο ... Ποίο είναι το σημαντικότερο στοιχείο που πρέπει να προσέξουμε σε αυτόν τον αλγόριθμο;

Scaling outline

```
for each x : 0 to destRect.w-1 do
  for each y : 0 to destRect.h-1 do {
    Let C = GetPixel(src, from.x + x / factor, from.y + y / factor);
    PutPixel(dest, to.x + x, to.y + y, C);
  }
```

```
for each x : 0 to destRect.w-1 do
  for each y : 0 to destRect.h-1 do {
    Let C = GetPixel(src, from.x + shrinkCoords[factor][x], from.y + shrinkCoords[factor][y]);
    PutPixel(dest, to.x + x, to.y + y, C);
  }
}
```

(ε) 5 Μονάδες. Συμπληρώστε τα κενά σημεία στην διατύπωση που περιγράφουν τη λογική της λειτουργίας light blit;

- Η λογική του light blit είναι να αποτυπώσει το κάθε pixel χωρίς της... pixel τόσο κοντά στο λευκό χρώμα (μέγιστο);όπως ορίζει η τιμή του alpha παραμέτρου.

- Η λογική του light blit είναι να αποτυπώσει το κάθε source pixel στο αντίστοιχο destination pixel τόσο κοντά στο λευκό χρώμα (μέγιστο φως) όσο ορίζει η τιμή του light παραμέτρου.

light value parameter L

Συμπληρώστε την παρακάτω εξίσωση που περιγράφει πως ένα source pixel P_s γίνεται rendered στο αντίστοιχο destination pixel P_d με light value $L = [0, 255]$

- Εάν $L = 0$ (δηλ. καθόλου φως), τότε $P_d = P_s$
- Εάν $L = 255$ (δηλ. maximum light), τότε $P_d = 255$ (λευκό)
- Αλλιώς, $P_d = P_s + (255 - P_s) * (L / 255)$