ΑΛΕΞΑΝΔΡΕΙΟ Τ.Ε.Ι. ΘΕΣΣΑΛΟΝΙΚΗΣ
ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΓΡΑΦΙΚΑ, ΠΑΙΓΝΙΔΙΑ ΚΑΙ ANIMATION ΜΕ JAVA

Του φοιτητή

Δημόπουλου Γρηγόρη

Αρ. Μητρώου: 06/3087

Επιβλέπων καθηγητής

Ράπτης Πασχάλης

Θεσσαλονίκη 2012

ΠΡΟΛΟΓΟΣ

Στα πλαίσια της πτυχιακής μου εργασίας στο τμήμα Πληροφορικής του ΑΤΕΙ Θεσσαλονίκης, μου ανατέθηκε η υλοποίηση της εργασίας με θέμα «Γραφικά, παιγνίδια και animation με Java».

Η εργασία αυτή αποτελείται από το πρακτικό και το θεωρητικό μέρος. Το θεωρητικό μέρος της εργασίας αποτελείται από την αναφορά των βιβλιοθηκών και των εφαρμογών της Java που χρησιμοποιούνται για την παραγωγή γραφικών, παιγνιδιών και animation. Επίσης έδωσα έμφαση στην ανάλυση του κώδικα των προγραμμάτων που σύνταξα για το πρακτικό μέρος της εργασίας, έτσι ώστε να υπάρχει μια ισορροπία και να μπορεί να γίνεται πιο κατανοητό. Ακόμα έγινε προσπάθεια έτσι ώστε κάθε έννοια να συνοδεύεται και από ένα ολοκληρωμένο και σύντομο παράδειγμα. Τα παραδείγματα είναι ελεγμένα και δεν έχουν κάποιο συντακτικό λάθος, όπως άλλωστε φαίνονται στις αντίστοιχες εικόνες οθονών (screenshots) που πρόσθεσα.

Το πρακτικό μέρος της εργασίας αποτελείται από ένα 2D παιχνίδι, ένα 3D πρόγραμμα γραφικών και μια σύντομης ταινίας animation. Το 2D παιχνίδι ονομάζεται Space Invaders, στο οποίο ο στόχος του παίκτη είναι να καταστρέψει όσο το δυνατόν περισσότερα εχθρικά αεροσκάφη, έτσι ώστε να εμποδίσει τους εξωγήινους να εισβάλουν στη Γη. Στο πρόγραμμα των γραφικών δίνεται η δυνατότητα στο χρήστη να μεγεθύνει, να μετακινήσει και να περιστρέψει κάποιο γεωμετρικό σχήμα ή ακόμα να αλλάξει το χρώμα ή την υφή αυτού. Με αυτόν τον τρόπο γίνεται κατανοητό, κατά κάποιον τρόπο, η σύνταξη και η κατασκευή πολύπλοκων 3D παιχνιδιών. Τέλος το σενάριο της ταινία animation εξελίσσεται έξω από μια τράπεζα, όπου ένας ανύποπτος πελάτης καθώς δέχεται την επίθεση ενός ληστή, προσπαθεί με την χρήση πολεμικών τεχνών να προστατευτεί.

ΠΕΡΙΛΗΨΗ

Η Πτυχιακή Εργασία «Γραφικά, παιγνίδια και animation με Java» εστιάζεται στην ανάλυση των βιβλιοθηκών, των μεθόδων και των κλάσεων της Java που είναι απαραίτητες για τη δημιουργία ενός παιχνιδιού, μιας animation ταινίας και ενός προγράμματος γραφικών. Έτσι λοιπόν ο αναγνώστης θα μάθει πως να δημιουργεί πλαίσια (JFrame) με τη Java Swing και να προσθέτει σε αυτά συστατικά όπως κουμπιά, ετικέτες, μενού μπάρες, επιλογέα αρχείων και επιλογέα χρωμάτων. Αυτά τα στοιχεία θα τον βοηθήσουν στον προγραμματισμό ενός απλού παιχνιδιού ή ενός προγράμματος γραφικών. Έπειτα γίνεται αναφορά στα νήματα τα οποία εφαρμόζονται σε κάθε applet ή εφαρμογή της Java. Τα νήματα παίζουν σημαντικό ρόλο στην ανάπτυξη των παιχνιδιών και των κινούμενων σχεδίων (animation) τα οποία χρειάζονται να εκτελούν πολλές ενέργειες ταυτόχρονα. Επίσης δίνεται έμφαση στην σύνταξη ενός animation με Java και τι εμπόδια συναντάμε κατά την εκτέλεση. Συγκεκριμένα δίνεται η λύση στο πρόβλημα του τρεμοπαιξίματος της οθόνης κατά την προβολή του animation.

Η ανάπτυξη ενός παιχνιδιού σε Java απαιτεί την ύπαρξη προγραμματισμένων κουμπιών όπου κάθε φορά που ο χρήστης πατάει ένα κουμπί να συμβαίνει το ανάλογο γεγονός. Στην πτυχιακή εργασία γίνεται ανάλυση των ακροατών για το χειρισμό των γεγονότων που παράγονται από τα κουμπιά του πληκτρολόγιου και του ποντικιού, καθώς και από την κίνηση του ποντικιού. Ακόμα περιέχει τις κατάλληλες μεθόδους για τον εμπλουτισμό του παιχνιδιού μας με ηχητικά εφέ και μουσική.

Στο τελευταίο κεφάλαιο της πτυχιακής εργασίας επικεντρώνεται στη Java 3D και τις βιβλιοθήκες και κλάσεις που περιλαμβάνει. Επίσης δίνονται διαγράμματα τύπου δέντρου τα οποία απεικονίζουν την ιεραρχία των κόμβων-αντικειμένων σε ένα γράφο σκηνής. Ακόμη περιγράφει αναλυτικά τις κλάσεις που χρειαζόμαστε για δημιουργήσουμε ένα γεωμετρικό σχήμα, να προσθέσουμε φωτισμό στο γράφημα, να τροποποιήσουμε την εμφάνιση του σχήματος, καθώς και να μετασχηματίσουμε τα σχήμα μας. Τέλος αναφέρεται στην κλάση η οποία ρυθμίζει το οπτικό πεδίο του χρήστη στο γράφο σκηνής και παρέχει μεθόδους για τον έλεγχο της κάμερας.

ABSTRACT

The thesis "Graphics, games and animation with Java» focuses on the analysis of libraries, methods and classes of Java, which are necessary for developing a game, an animation film and a graphics program. So the reader will learn how to create frames (JFrame) with Java Swing and add components on them like buttons, labels, menu bars, file chooser and color chooser. These data will assist in planning a simple game or a graphics program. Then there is a reference to the threads that are applied to each applet or application of Java. The threads have an important role in the development of games and animation which need to execute multiple actions simultaneously. It's also given emphasis on writing an animation with Java and what obstacles we encounter during the process. It's specifically given the solution to the problem of screen flickering when viewing the animation.

Developing a game in Java requires the existence of programming buttons in which every time the user presses a button, it produces the relevant event. In this thesis quoted an analysis of listeners about the control of events that are generated by keyboard and mouse buttons, and the mouse movement. It also includes appropriate methods for enriching our game with sound effects and music.

The last chapter of thesis focuses on Java 3D and libraries and classes that includes. It's also given tree type diagrams, which illustrate the hierarchy of nodes-objects in a scene graph. It also describes the classes that we need to construct a geometric shape, add lighting in the object, modify the appearance of the figure, and to transform our figure. Finally, it's referred to the class which adjusts the user's view in scene graph and provides methods for controlling the camera view.

Πίνακας περιεχομένων

Ευρετήριο σχημάτων

ΕΙΣΑΓΩΓΗ

ΤΙ ΕΙΝΑΙ Η JAVA;

Η Java είναι μία αντικειμενοστραφής γλώσσα προγραμματισμού, η οποία σχεδιάστηκε στις αρχές του 1991 από την εταιρία πληροφορικής Sun Microsystems. Η Java χρησιμοποιείται για την ανάπτυξη λογισμικού σε μικρο-συσκευές έως και πολύπλοκα συστήματα παραγωγής γραφικών. Αν τη συγκρίνουμε από πλευρά δομής με τις άλλες γλώσσες προγραμματισμού, θα μπορούσαμε να πούμε ότι η Java μοιάζει με περισσότερο με τη C. Της μοιάζει αρκετά στον τρόπο σύνταξης και φιλοσοφίας, αλλά δεν είναι C.

ΤΑ ΠΛΕΟΝΕΚΤΗΜΑΤΑ ΤΗΣ JAVA

Υπάρχουν κάποια χαρακτηριστικά τα οποία κάνουν την Java ξεχωρίζει από τις άλλες γλώσσες. Αυτά είναι:

- Είναι απλή. Αυτό επειδή οι δημιουργοί της Java απέκλεισαν από την γλώσσα κάποια δύσκολα χαρακτηριστικά τα οποία συνήθως συναντάμε σε άλλες γλώσσες όπως τη C.
- Είναι αντικειμενοστραφής γλώσσα προγραμματισμού. Η Java χρησιμοποιεί τις κλάσεις για να οργανώσει τον κώδικα της σε ενότητες. Κατά την εκτέλεση, το πρόγραμμα δημιουργεί τις κλάσεις-αντικείμενα. Τα αντικείμενα αποτελούνται από πεδία και μεθόδους. Τα πεδία περιγράφουν τι είναι το αντικείμενο, ενώ οι μέθοδοι περιγράφουν τι κάνει το αντικείμενο. Επίσης οι κλάσεις μπορούν να κληρονομήσουν ιδιότητες από άλλες κλάσεις, αλλά δεν επιτρέπεται η πολλαπλή κληρονομικότητα, στην οποία η κλάση δεν επιτρέπεται να κληρονομήσει πεδία και μεθόδους από περισσότερες από μία άλλες κλάσεις.
- Είναι ασφαλής. Η Java έχει σχεδιαστεί με τρόπο ώστε να παρέχει ασφαλή εκτέλεση του κώδικα στο δίκτυο. Για παράδειγμα δεν περιέχει δείκτες (pointers), ούτε μπορεί να γίνει αυθαίρετη προσπέλαση διευθύνσεων της μνήμης. Έτσι τα μέτρα αυτά παρέχουν προστασία ενάντια στους ιούς.

- Είναι γλώσσα υψηλού επιπέδου (high level language). Αυτό σημαίνει ότι οι εντολές της είναι λέξεις, τις οποίες καταλαβαίνει μεν ο άνθρωπος, αλλά όχι και ο υπολογιστής. Για να μπορέσει να τις καταλάβει ο υπολογιστής χρειαζόμαστε έναν μεταγλωττιστή (compiler). Ο μεταγλωττιστής είναι ένα πρόγραμμα το οποίο μετατρέπει ολόκληρο τον κώδικα υψηλού επιπέδου σε κώδικα γλώσσας μηχανής πριν την εκτέλεση του.
- Τα προγράμματα σε Java μπορούν τα τρέξουν σε οποιαδήποτε λειτουργικό σύστημα. Αυτό γίνεται διότι ο μεταγλωττιστής της Java δε δημιουργεί γηγενή κώδικα προσαρμοσμένο στο συγκεκριμένο τύπο υπολογιστή, αλλά δημιουργεί κώδικα byte. Έτσι ένα πρόγραμμα που έχει δημιουργηθεί σε ένα υπολογιστή με Windows μπορεί να τρέξει χωρίς κανένα πρόβλημα σε ένα υπολογιστή με λειτουργικό σύστημα Unix.
- Υποστηρίζει πολυνημάτωση (multithreding). Ένα πρόγραμμα Java μπορεί να περιλαμβάνει πολλές ξεχωριστές διεργασίες, οι οποίες να εκτελούνται συνεχώς και ανεξάρτητα η μια από την άλλη.
- Τα προγράμματα της Java κάνουν από μόνα τους περισυλλογή «απορριμμάτων» (garbage collection). Αυτό σημαίνει ότι ο προγραμματιστής δεν χρειάζεται να φροντίσει πλέον ο ίδιος για τη διαγραφή άχρηστων δεδομένων από τη μνήμη.
- Μπορεί να δημιουργήσει γρήγορο κώδικα. Επειδή η χρήση του διερμηνευτή και του μεταγλωττιστή θεωρείται σχετικά αργή, οι εταιρίες ανέπτυξαν ειδικούς μεταγλωττιστές οι οποίοι μετατρέπουν τον κώδικα byte σε κώδικα γλώσσας μηχανής πολύ γρήγορα. Οι μεταγλωττιστές αυτοί ονομάζονται “just in time compilers”.
- Παρέχει ένα μεγάλο αριθμό βιβλιοθηκών κώδικα για διάφορες χρήσεις, όπως δημιουργία 3D γραφικών, μαθηματικές πράξεις, προσπέλαση αρχείων, χειρισμό σχεσιακών βάσεων δεδομένων κ.λπ.
- Η Java μας δίνει τη δυνατότητα να δημιουργήσουμε δυναμικές ιστοσελίδες (dynamic web pages).

ΤΑ ΜΕΙΟΝΕΚΤΗΜΑΤΑ ΤΗΣ JAVA

Πέραν τα πλεονεκτήματα όμως, που έχει η Java, υπάρχουν και μειονεκτήματα. Αυτά είναι:

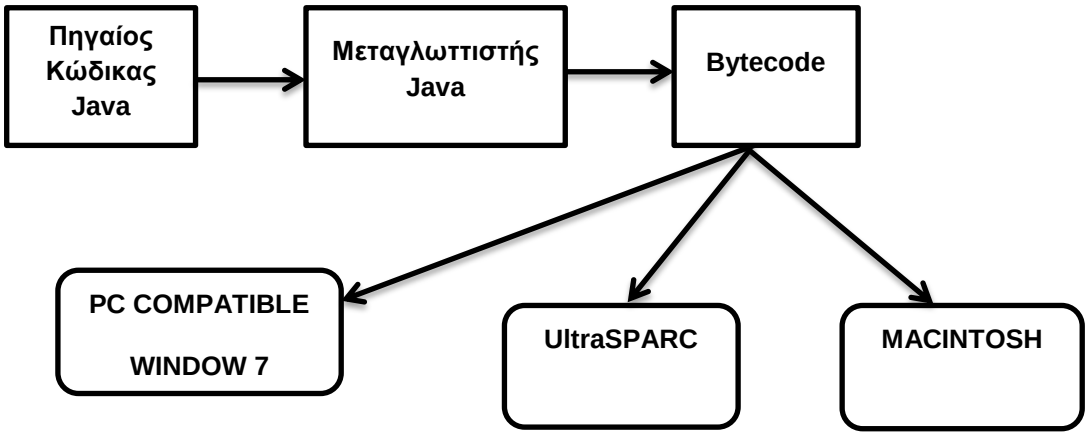
- Τα προγράμματα πλοήγησης ενδέχεται να μην μπορούν να εκτελέσουν την μικροεφαρμογή Applet μας. Αυτό διότι τα προγράμματα πλοήγησης ενδέχεται να μην είναι εφοδιασμένα με την τρέχουσα έκδοση της Java.
- Η Java αρχικά ήταν πιο αργή σε σχέση με άλλες προγραμματιστικές γλώσσες υψηλού επιπέδου (high-level) όπως η C και η C++. Όπως όμως αναφέραμε παραπάνω στα πλεονεκτήματα με καθιέρωση των μεταγλωττιστών JIT (Just In Time), οι οποίοι μετατρέπουν τον κώδικα byte απευθείας σε γλώσσα μηχανής, η διαφορά ταχύτητας από τη C++ έχει μικρύνει κατά πολύ.

Ο ΔΙΕΡΜΗΝΕΥΤΗΣ ΚΑΙ Ο ΜΕΤΑΓΛΩΤΤΙΣΤΗΣ ΤΗΣ JAVA

Όπως αναφέραμε παραπάνω στα πλεονεκτήματα η Java διαφέρει από τις άλλες γλώσσες προγραμματισμού ως προς την μετατροπή του πηγαίου κώδικα σε κώδικα μηχανής. Σε μια παραδοσιακή γλώσσα (Pascal, C, κ.λπ.) ο μεταγλωττιστής μετατρέπει τον κώδικα του προγράμματος που έχουμε γράψει σε δυαδικό αρχείο (binary file), δηλαδή σε κώδικα μηχανής κατάλληλο για τον συγκεκριμένο επεξεργαστή. Στη Java όμως είναι διαφορετικά τα πράγματα. Η Java μπορεί να δημιουργεί πρόγραμμα το οποίο να μπορεί να τρέχει σε μια ποικιλία υπολογιστών και λειτουργικών συστημάτων (Σχήμα 1).

Το περιβάλλον ανάπτυξης περιλαμβάνει δύο μέρη:

1. Το μεταγλωττιστή, ο οποίος αντί για δυαδικό αρχείο δημιουργεί ένα είδος κώδικα που λέγεται κώδικας byte και είναι ανεξάρτητος από πλατφόρμα.
2. Το διερμηνευτή κώδικα byte, ο οποίος ονομάζεται και Εικονική Μηχανή Java (Java Virtual Machine, JVM) ή Διερμηνευτής Java Χρόνου (Java Runtime Interpreter). Ο διερμηνευτής διαβάζει τον κώδικα byte και εκτελεί τις κατάλληλες εντολές γλώσσες μηχανής που αντιστοιχούν στον κατάλληλο επεξεργαστή.



Σχήμα 1 "Η επεξεργασία του πηγαίου κώδικα Java"

JAVA APPLETS

Ένα κύριο στοιχείο που κάνει τη Java να ξεχωρίζει από τις άλλες γλώσσες προγραμματισμού, είναι ότι δίνει τη δυνατότητα στον προγραμματιστή να δημιουργήσει κάποια ειδικά προγράμματα, τα οποία λέγονται applets (μικροεφαρμογές). Τα Java Applets εμφανίστηκαν το 1995 στην πρώτη έκδοση της γλώσσας Java και γράφτηκαν σε γλώσσες προγραμματισμού τα οποία μεταγλωττιζόντουσαν σε Java bytecode, συνήθως σε Java, καθώς επίσης και σε άλλες γλώσσες όπως Jython, JRuby, ή Eiffel. Επιπλέον τα Java Applets τρέχουν πολύ γρήγορα σε γενικές γραμμές, αλλά γενικά πιο αργά από άλλες γλώσσες όπως η C++. Γενικά τα Applets χρησιμοποιούνται για να παρέχουν χαρακτηριστικά στις διαδικτυακές εφαρμογές στις οποίες ο HTML δεν μπορεί να παρέχει. Αυτά έχουν την ικανότητα να συλλαμβάνουν την κίνηση του ποντικιού και επίσης έχουν χειρισμό όπως κουμπιά ή κουτιά επιλογής. Σε απάντηση της ενέργειας του χρήστη το applet μπορεί να αλλάξει κάποιο γραφικό περιεχόμενο ή να γίνει κάποια κίνηση ενός αντικειμένου. Τα Applets επίσης χρησιμοποιούνται για την κατασκευή δικτυακών παιχνιδιών, τα οποία δίνουν το δικαίωμα στον παίχτη να αγωνιστεί με κάποιον άλλο αληθινό αντίπαλο μέσω του διαδικτύου.

ΓΙΑΤΙ JAVA ΓΙΑ ΤΗΝ ΑΝΑΠΤΥΞΗ ΠΑΙΧΝΙΔΙΩΝ;

Είναι η Java μια αξιόπιστη γλώσσα για αναπτύξουμε παιχνίδια με αυτή; Στην αρχή του κεφαλαίου τονίσαμε ότι η Java έχει πολλά πλεονεκτήματα σε σύγκριση με άλλες γλώσσες προγραμματισμού, όπως ότι είναι απλή, ασφαλής, αντικειμενοστραφής και υψηλού επιπέδου γλώσσα προγραμματισμού, τα προγράμματα της τρέχουν σε οποιαδήποτε λειτουργικό σύστημα κ.λπ. Όμως για την ανάπτυξη παιγνιδιών σε Java υπάρχουν κάποιες ενστάσεις για την καταλληλότητα της. Αυτές είναι οι εξής:

1. Η Java είναι πολύ αργή για ανάπτυξη παιχνιδιών.
2. Η Java έχει διαρροή μνήμης.
3. Η Java είναι πολύ υψηλού επιπέδου γλώσσα.
4. Η εγκατάσταση των εφαρμογών σε Java είναι αρκετά δύσκολη.
5. Η Java δεν υποστηρίζεται από τις κονσόλες παιχνιδιών.
6. Κανένας δεν χρησιμοποιεί την Java για να αναπτύξει απαιτητικά παιχνίδια.
7. Η Sun Microsystems δεν ενδιαφέρεται να υποστηρίξει και να προωθήσει παιχνίδια σε Java.

Αξίζει να σημειώσω ότι σχεδόν όλα τα μειονεκτήματα είναι ουσιαστικά λάθος. Η Java σε γενικές γραμμές έχει την ίδια ταχύτητα με την C++. Οι διαρροές μνήμης μπορούν να αποφευχθούν με καλό προγραμματισμό και τεχνικές. Όντως η Java είναι υψηλού επιπέδου γλώσσα, αλλά προσφέρει καλύτερη απευθείας πρόσβαση στο υλικό και τις εξωτερικές συσκευές γραφικών. Η εγκατάσταση δεν είναι δύσκολη αν χρησιμοποιούμε ευπρεπή λογισμικό εγκατάστασης. Υπάρχει ένας αυξανόμενος αριθμός από υπέροχα και διασκεδαστικά παιχνίδια σε Java και μία τεράστια υποστήριξη από την Sun και τις ιστοσελίδες της Sun.

Το γενικό νόημα των αντιρρήσεων που υπάρχουν στην ανάπτυξη παιχνιδιών σε Java είναι ότι είχαν μεγαλύτερη ισχύ στα τέλη της δεκαετίας του 90, όταν η γλώσσα και οι βιβλιοθήκες της ήταν λιγότερο πολύπλοκες και αργές. Οι χρήστες της Java και οι προγραμματιστές της αυξάνονται και έχουν παραγάγει μια πληθώρα από χρήσιμα εργαλεία, διαδικτυακή βοήθεια και παραδείγματα με κώδικα. Επίσης αναπτυχτήκαν τόποι δημόσιας συζήτησης (forums) στο διαδίκτυο

με θέμα τα παιχνίδια σε Java, ενώ τα τελευταία χρόνια ούτε καν υπήρχαν αυτά. Τέλος θεωρώ τη Java εξαιρετική γλώσσα για ανάπτυξη παιχνιδιών.

Παρακάτω θα γίνει ανάλυση στα μειονεκτήματα που έχει η Java στην ανάπτυξη παιχνιδιών.

1. Η Java είναι πολύ αργή για ανάπτυξη παιχνιδιών

Αυτός ο τίτλος θα μπορούσε να αναδιατυπωθεί ως «Η Java είναι αργή σε σύγκριση με τη C και C++, οι οποίες είναι οι γλώσσες που κυριαρχούν στον προγραμματισμό παιχνιδιών». Αυτός ο ισχυρισμός υπήρξε όταν πρωτοεμφανίστηκε η Java (περίπου το 1996) αλλά ουσιαστικά η διαφορά ταχύτητας μεταξύ Java και C/C++ μειώνεται σημαντικά κάθε φορά που βγαίνει μια νέα έκδοση. Κάποια στατιστικά στοιχεία είχαν αποδείξει ότι η JDK 1.0, η οποία ήταν η πρώτη έκδοση της γλώσσας, ήταν 20 με 40 φορές πιο αργή από την C++. Παρόλα αυτά, η τωρινή έκδοση J2SE 5.0 έχει μικρύνει εμφανέστατα την διαφορά στο 1,1 φορές πιο αργή.

Όλοι αυτοί οι αριθμοί φυσικά εξαρτώνται και από την ικανότητα που έχει ένας προγραμματιστής να αναπτύσσει ένα πρόγραμμα αποτελεσματικά, καθώς επίσης και στο στυλ κώδικα που χρησιμοποιεί. Ωστόσο η επιτάχυνση της Java έχει στην πραγματικότητα γίνει χάρη στη βελτίωση των μεταγλωττιστών. Η τεχνολογία Hotspot παρουσιάστηκε από την J2SE 1.3 και στόχος της ήταν να εντοπίζει κατά την εκτέλεση τις κρίσιμες περιοχές του κώδικα, οι οποίες χρησιμοποιούνταν πολλές φορές και ήταν δύσκολο να μεταγλωττιστούν. Η Hotspot τεχνολογία είναι σχετικά καινούργια και πιθανότατα οι νέες εκδόσεις της Java στο μέλλον να αποφέρουν μεγαλύτερη επιτάχυνση.

2. Η Java έχει διαρροή μνήμης

Όταν οι προγραμματιστές της C/C++ αναφέρουν ότι η Java έχει διαρροή μνήμης πιθανότατα δεν καταλαβαίνουν πως η Java λειτουργεί. Η Java δεν έχει αριθμητικούς δείκτες (τυπική διαρροή μνήμης στη C), καθώς επίσης δεν έχει πρόσβαση σε out-of-bounds πίνακες οι οποίοι κόβονται από τον μεταγλωττιστή.

Ωστόσο, αυτοί οι προγραμματιστές μπορεί να εννοούν ότι τα αντικείμενα τα οποία δεν χρειάζονται από το πρόγραμμα, δεν μαζεύονται από τον συλλέκτη απορριμμάτων. Αυτό το πρόβλημα παρουσιάζεται όταν το πρόγραμμα δημιουργεί συνέχεια νέα αντικείμενα και δεσμεύει πολύ μνήμη με αποτέλεσμα τελικά να τερματίσει (crash) το πρόγραμμα λόγω έλλειψης ελεύθερης μνήμης. Είναι σημαντικό να πούμε ότι αυτό του είδους το πρόβλημα είναι συνέπεια κακού προγραμματισμού.

Άλλο ένα ακόμα πρόβλημα σχετικό με τη μνήμη που έχει αναφερθεί είναι ο συλλέκτης απορριμμάτων της Java εκτελείται σε μικρά χρονικά διαστήματα, προκαλώντας διακοπή της εφαρμογής για λίγα δευτερόλεπτα, την ώρα που ο συλλέκτης σκουπίζει και καθαρίζει. Η Java Virtual Machine (JVM) διατίθεται με κάμποσους διαφορετικούς συλλέκτες απορριμμάτων, οι οποίοι συλλέγουν με ποικίλους τρόπους καθώς και να επιλεχτούν και να συντονιστούν μέσω του command line.

3. Η Java είναι πολύ υψηλού επιπέδου γλώσσα

Αυτό το μειονέκτημα είναι το παλιότερο χρονολογικά σε σχέση με την ταχύτητα της Java και τον έλεγχο της μνήμης που αναφερθήκαμε παραπάνω. Οι λεπτομέρειες σχετικά με το ζήτημα αυτό είναι οι εξής:

1) Η χρήση των κλάσεων, των αντικειμένων και της κληρονομικότητας που κάνει η Java προσθέτει πολλές δαπάνες στο πρόγραμμα χωρίς να υπάρχει κάποιο ουσιαστικό όφελος στον κώδικα.
2) Η ανεξαρτησία μηχανής που προσφέρει η Java πολλές φορές σημαίνει ότι χαμηλού επιπέδου γρήγορες λειτουργίες, για παράδειγμα direct VIDEO RAM I/O, είναι αδύνατες.

Το ζήτημα 1 αγνοεί τα προφανή οφέλη της επαναχρησιμοποίησης και της επέκτασης που παρέχει η μεγάλη βιβλιοθήκη της Java, η οποία περιλαμβάνει υψηλής ταχύτητας I/O, προηγμένα 2D και 3D γραφικά και πολλές δικτυακές τεχνικές. Επίσης ξεχασμένα είναι τα πλεονεκτήματα του αντικειμενοστραφούς

προγραμματισμού, που εμπεριέχουν UML, τα οποία κάνουν τα σύνθετα, μεγάλα και πραγματικά συστήματα πιο ελέγξιμα κατά τη διάρκεια της ανάπτυξης, της εφαρμογής και της συντήρησης.

Το ζήτημα 2 έχει επίπτωση στα παιχνίδια όταν εξετάζουμε υψηλής ταχύτητας γραφικά, αλλά αυτό απευθύνεται σε πρόσφατες εκδόσεις της Java. Η J2SE 1.4 παρουσίασε τη αποκλειστική μέθοδος πλήρης οθόνης (full-screen exclusive mode, FSEM), η οποία ανέστελλε το κανονικό περιβάλλον με τα παράθυρα και επέτρεπε στην εφαρμογή να έχει απευθείας πρόσβαση στα υλικό των γραφικών. Ακόμα αυτό επιτρέπει τεχνικές όπως γύρισμα σελίδων, και παρέχει έλεγχο στην ανάλυση της οθόνης και στο βάθος της εικόνας. Τέλος ο κυριότερος στόχος του FSEM είναι να επιταχύνει τις εφαρμογές γραφικών, όπως τα παιχνίδια.

4. Η εγκατάσταση των εφαρμογών σε Java είναι αρκετά δύσκολη

Κάποιοι ισχυρίζονται ότι θα πρέπει να είσαι ειδικός στην Java για να εγκαταστήσεις και να εκτελέσεις μια εφαρμογή, επειδή τα περισσότεροι παίκτες παιχνιδιών θέλουν να κάνουν λίγα κλικ και επιλογές για να ξεκινήσουν ένα παιχνίδι. Παρακάτω θα δούμε περισσότερα σχόλια σχετικά με αυτό το θέμα. Αυτά είναι:

1) Η Java (πιο συγκεκριμένα η JRE) πρέπει να είναι εγκατεστημένη στο σύστημα πριν εκτελέσουμε την εφαρμογή.
2) Ο κώδικας δυσχεραίνει το σύστημα. Ακόμα και μικρά προγράμματα χρειάζονται ένα 15MB JRE.
3) Αλλάζοντας συχνά το JVMs κάνει δύσκολο να γράψεις κώδικα ο οποίος θα δουλεύει για κάθε πιθανή έκδοση της Java.
4) Ασυνήθη στοιχεία είναι συνήθως απαραίτητα, π.χ. η Java 3D προκαλεί ακόμα περισσότερα προβλήματα εγκατάστασης.
5) Είναι αδύνατο να μεταγλωττίσεις μια εφαρμογή για μια συγκεκριμένη πλατφόρμα.
6) Τα αρχεία με επέκταση .jar έχουν συνήθως καταλειφθεί από άλλο λογισμικό (π.χ. από συμπιεσμένα προγράμματα) την ώρα της

εκτέλεσης. Αυτό σημαίνει ότι ο χρήστης δεν μπορεί να κάνει διπλό κλικ σε ένα JAR αρχείο για να ξεκινήσει.

7) Το JRE είναι πιο αργό στην εκκίνηση σε σύγκριση με μια μεταγλωττισμένη εφαρμογή.

Όλα αυτά τα προβλήματα, εκτός ίσως το 2 και το 7, μπορούν να λυθούν με τη χρησιμοποίηση ενός καλού λογισμικού. Υπάρχουν 2 λύσεις σχετικά με την εγκατάσταση μιας εφαρμογής.

5. Η Java δεν υποστηρίζεται από τις κονσόλες παιχνιδιών

Δυστυχώς αυτή η κριτική δικαιολογείται. Τα βιντεοπαιχνίδια ουσιαστικά είναι μια βιομηχανία πολλών δισεκατομμυρίων δολαρίων. Οι εταιρίες βιντεοπαιχνιδιών πουλάνε κάθε χρόνο ένα μεγάλο αριθμό παιχνιδιών, επιτυγχάνοντας το 2011 εισόδημα περίπου στα 25 δισεκατομμύρια δολάρια. Το αρνητικό είναι ότι μόνο το 10%-20% των παιχνιδιών αυτό βγαίνουν σε PCs. Η πλειοψηφία των παιχνιδιών βγαίνουν για τις 3 κονσόλες: Playstation 3 (PS3), XBOX 360 και Nintendo Wii. Η Sony έχει κυριαρχήσει στον αριθμό κονσόλων που έχει κατασκευάσει, έχοντας πουλήσει περίπου διπλάσιες κονσόλες σε σύγκριση με τη Microsoft και τη Nintendo μαζί.

Ένα ακόμα αρνητικό φαινόμενο είναι ότι η Java δεν είναι διαθέσιμη στο Playstation. Το πρόβλημα αυτό το έχει αναγνωρίσει η Sun και έχει δείξει τη διάθεση της, μέσω συνεδρίων με τη Sony, να υπάρξει κάποια συνεργασία στο μέλλον.

Η Java στο μέλλον ίσως να έχει κάποια καλύτερη ευκαιρία και να γίνουν αποδεκτές από τις εταιρίες κατασκευής κονσόλων. Αυτό μπορεί να γίνει επειδή υπάρχουν 2 τάσεις. Πρώτον οι κονσόλες τείνουν να γίνουν συσκευές σπιτιού και δεύτερον η μεγάλη αύξηση των διαδικτυακών παιχνιδιών. Και οι δύο τάσεις αναγκάζουν τις κονσόλες να προσφέρουν σύνθετο δίκτυο και υποστήριξη από το διακομιστή, δυο πράγματα που μπορεί να καλύψει εύκολα η Java και η Sun.

6. Κανένας δεν χρησιμοποιεί την Java για να αναπτύξει απαιτητικά παιχνίδια

Η αλήθεια είναι ότι ο αριθμός των εμπορικών παιχνιδιών γραμμένα σε Java είναι μικρός σε σύγκριση με αυτά της C ή C++, αλλά αυτός ο αριθμός αυξάνεται κάθε χρόνο και μερικά από αυτά πήραν βραβεία και είχαν επιτυχία στις πωλήσεις. Μερικά από τα παιχνίδια αυτά είναι:

- Puzzle Pirates της Three Rings (http://www.puzzlepirates.com/)
- Law and Order II της Legacy Interactive (http://www.lawandordergame.com/index2.htm)
- Kingdom of Wars της Abandon Castle Studios (http://www.abandonedcastle.com/)
- Alien Flux της Puppy Games (http://www.puppygames.net/info.php?game=Alien_Flux)
- Pernica της Starfire Research (http://www.starfireresearch.com/pernica/pernica.html)
- FlyingGuns (http://www.flyingguns.com/)
- Out of Space (http://www.geocities.com/Psionic1981)
- CazaPool3D (http://cazapool3d.sourceforge.net/cazapooljws/Pool.html)

7. Η Sun Microsystems δεν ενδιαφέρεται για τα Java Games

Η Sun δεν έχει παράδοση στην υποστήριξη και ανάπτυξη παιχνιδιών, και πιθανότατα να μην αποκτήσει ποτέ τη γνώση που έχει η Sony ή η Nintendo. Παρ' όλα αυτά, τα τελευταία χρόνια η Sun έχει δείξει μεγαλύτερη αφοσίωση στα παιχνίδια.

Η Java 3D είναι ένα κοινό project του Advanced Software Development Group της Sun με ότι αυτό συνεπάγεται με όρους τεχνολογικής επάρκειας, πρωτοπορίας, καθώς και δέσμευσης για εξέλιξη. Βασικά υπάρχει ένα group ειδικών που καθορίζει, σχεδιάζει και υλοποιεί τις διάφορες εκδόσεις της Java 3D, με ένα σημαντικό κομμάτι της υλοποίησης να γίνεται από την κοινότητα, και μία

στρατηγική που εφαρμόστηκε με επιτυχία στην ανάπτυξη των JOGL, JOAL, και JInput (APIs).

Υπάρχουν τέσσερις Java 3D Mailing lists:

- interest@java3d.dev.java.net
- announce@java3d.dev.java.net
- issues@java3d.dev.java.net
- cvs@java3d.dev.java.net

Υπάρχει ένα Java 3D Desktop Forum στο:

http://www.javadesktop.org/forums/forum.jspa?forumID=55

Άλλες πηγές σχετικά με τη Java 3D δίνονται ενδεικτικά παρακάτω:

- Η σελίδα Java 3D Product Page (http://java.sun.com/products/java-media/3D/), με συνδέσμους που παραπέμπουν σε demos, μια σελίδα με FAQs και πολλές άλλες εφαρμογές.
- Το Java 3D Gaming Forum (http://www.javagaming.org/cgi-bin/JGNetForums/YaBB.cgi?board=3D).
- Το Java Technology Forum για την Java 3D (http://forum.java.sun.com/forum.jsp?forum=21).
- Η καλύτερη ίσως ανεξάρτητη ιστοσελίδα για την Java 3D είναι η (http://www.j3d.org). Αυτή προσφέρει μία πολύ καλή σελίδα με FAQs, καθώς και μία μεγάλη συλλογή παραδειγμάτων, βοηθημάτων και κώδικα.

ΚΕΦΑΛΑΙΟ 1

JAVA SWING ΚΑΙ JAVA 2D

ΕΙΣΑΓΩΓΗ

Η Swing είναι το κύριο εργαλείο της Java για την δημιουργία γραφικού περιβάλλοντος επικοινωνίας (GUI). Η Swing είναι μέρος της Java Foundation Classes (JFC), καθώς επίσης και η AWT (Abstract Window Toolkit) βιβλιοθήκη, η οποία ανήκει στην Oracle. Η JFC είναι ουσιαστικά μία εφαρμογή προγραμματισμού διεπιφανειών (Application Programming Interface, API) η οποία παρέχει ένα γραφικό περιβάλλον επικοινωνίας για τα Java προγράμματα.

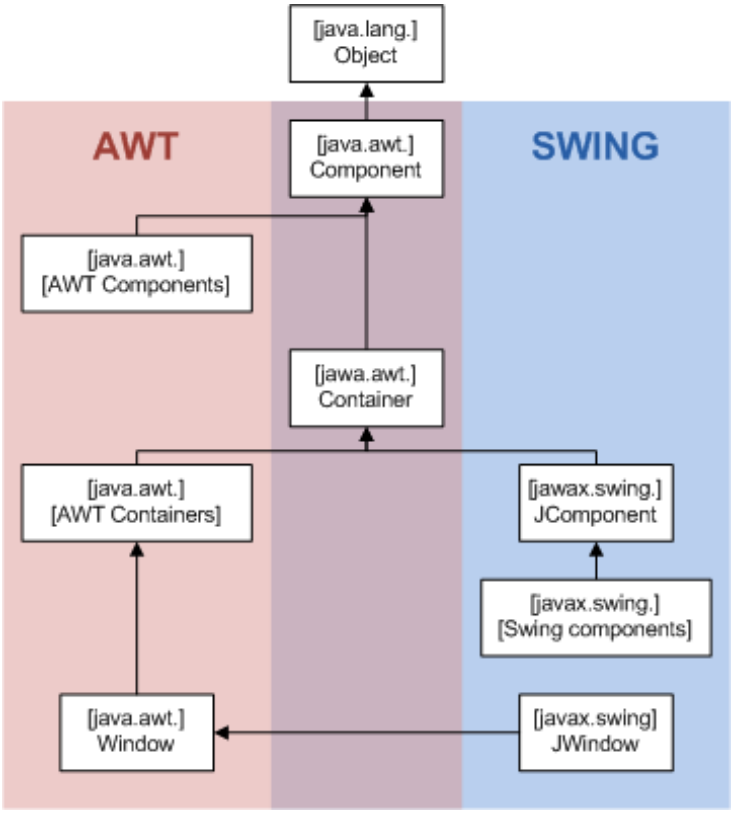
Η Swing περιλαμβάνει πάρα πολλά συστατικά για τη δημιουργία γραφικού περιβάλλοντος επικοινωνίας (GUI), πολύ περισσότερα από τη βιβλιοθήκη πακέτων της AWT (Abstract Window Toolkit). Δεν θα μπορούσε όμως να χαρακτηριστεί ως αντικαταστάτης του AWT, αλλά μάλλον ως μια ουσιώδης επέκτασή του. Άλλωστε χρησιμοποιεί και αυτός τους ίδιους διαχειριστές διάταξης, καθώς και το ίδιο μοντέλο χειρισμού γεγονότων.

Είναι σημαντικό να τονίσουμε ότι η Swing περιλαμβάνει διάφορα συστατικά τα οποία είναι κουμπιά, ετικέτες, πλαίσια ελέγχου, τμήματα παραθύρων με δυνατότητα κύλισης, περιγράμματα, μενού, γραμμές εργαλείων, λεζάντες κ.λπ. Το κάθε όνομα των συστατικών της Swing θα πρέπει να ξεκινάει με το γράμμα J και να ακολουθεί το όνομα του συστατικού με το πρώτο γράμμα να είναι κεφαλαίο. Για παράδειγμα αν θέλουμε να δηλώσουμε μία ετικέτα θα πρέπει να γράψουμε: JLabel <όνομα_ετικέτας>.

Τα περισσότερα χαρακτηριστικά της Swing είναι «ελαφρά» (Lightweight) με την έννοια ότι είναι όλα γραμμένα σε γλώσσα Java, σε αντίθεση με τα βαριά (Heavyweight) αντίστοιχα χαρακτηριστικά του AWT που είναι γραμμένα με γηγενή κώδικα για να μπορούν να ζωγραφιστούν στην οθόνη του συγκεκριμένου λειτουργικού συστήματος. Αυτό τους δίνει το δικαίωμα να αλλάζουν μορφή ακόμα και με δυναμικό τρόπο. Αν ο χρήστης που γράφει την εφαρμογή του σε Windows θέλει τα συστατικά της να εμφανίζονται π.χ. στο Motif της Sun, έχει τη δυνατότητα

να το κάνει έτσι. Μοναδική εξαίρεση αποτελούν τα συστατικά JApplet, JFrame, Jwindow και JDialog τα οποία είναι βαριά συστατικά.

Στο Σχήμα 2 βλέπουμε την ιεραρχία κλάσεων του AWT και της Swing:



Σχήμα 2 "Ιεραρχία κλάσεων AWT και Swing"

Η JFC που αναφερθήκαμε παραπάνω είναι ένα ευρύτερο σύνολο βιβλιοθηκών κλάσεων, που περιλαμβάνει μεταξύ άλλων και:

- Τα συστατικά γραφικού περιβάλλοντος διασύνδεσης «ελαφρού» τύπου, δηλαδή τη Swing.
- Τη βιβλιοθήκη γραφικών συστατικών «βαρέως» τύπου (AWT).
- Το μοντέλο αποστολής γεγονότος (Delegation Event Model).
- Δυνατότητα «μεταφοράς και απόθεσης» (drag and drop).
- Λειτουργίες γραφικών 2 διαστάσεων (Java 2D).
- Διεθνοποίηση (Internationalization).
- Βιβλιοθήκες κλάσεων για ειδικές λειτουργίες όπως μεγέθυνση περιεχομένου, ανάγνωση οθόνης, αναγνώριση ομιλίας κ.λπ. οι οποίες είναι ειδικά χρήσιμες σε άτομα με ειδικές ανάγκες.

Η Swing αποτελείται από πολλά πακέτα τα οποία περιλαμβάνουν πολλές κλάσεις και διασυνδέσεις. Κάποια σημαντικά πακέτα είναι:

- javax.swing – είναι το κύριο πακέτο όλων των άλλων που χρησιμοποιούνται στο Swing. Περιέχει κλάσεις και διασυνδέσεις για GUI συστατικά.
- javax.swing.event – ορίζει τα γεγονότα και ποιοι θα χρησιμοποιήσουν τα συστατικά της Swing.
- javax.swing.filechooser – έχει κλάσεις και διασυνδέσεις για την υποστήριξη του συστατικού διαλόγου JFilechooser.
- javax.swing.table - παρέχει κλάσεις και διασυνδέσεις για την υποστήριξη δεδομένων με μορφή πίνακα.
- javax.swing.text – περιλαμβάνει κλάσεις και διασυνδέσεις για την υποστήριξη των συστατικών κειμένου του Swing.

Η υπερκλάση όλων των «ελαφρών» συστατικών της Swing είναι η κλάση JComponent. Η JComponent έχει ένα μεγάλο αριθμό υποκλάσεων. Μερικές από αυτές είναι:

- AbstractButton – είναι η ανώτερη κλάση κουμπιών και έχει τις υποκλάσεις: JButton, JMenuItem και JToggleButton.
- JLabel – ορίζει μια ετικέτα στην οποία μπορεί να προσθέσουμε κάποιο κείμενο, εικόνα ή και τα δύο.
- JMenuBar – δημιουργεί μία ράβδο μενού στο πάνω μέρος του παραθύρου.
- JOptionPane – εμφανίζει ένα αναδυόμενο παράθυρο διαλόγου το οποίο μπορεί να ζητάει το χρήστη να εισάγει μια τιμή ή να τον πληροφορεί για κάτι.
- JPopupMenu – είναι ένα αναδυόμενο μενού το οποίο περιέχει κείμενο και γραφικά
- JScrollBar – δημιουργεί μια ράβδο κύλισης.
- JTable – δημιουργεί ένα πίνακα που μπορεί να εισάγουμε στα κελιά του κείμενο ή γραφικά.
- JTextComponent – είναι μια υπερκλάση για τα πεδία κειμένου του Swing.

ΥΠΟΚΕΦΑΛΑΙΟ
1.1 ΔΗΜΙΟΥΡΓΙΑ ΕΝΟΣ ΑΠΛΟΥ ΠΑΙΧΝΙΔΙΟΥ ΜΕ JAVA SWING

Σε αυτό το παράδειγμα θα δημιουργήσουμε ένα απλό παιχνίδι με τη χρήση της υποκλάσης JOptionPane η οποία είναι συστατικό της Swing όπως αναφέραμε παραπάνω. Ο στόχος του παίκτη είναι να μαντέψει ένα νούμερο από το 1 έως το 100 έχοντας το δικαίωμα να ξαναπροσπαθήσει αν έχει κάνει λάθος. Ουσιαστικά αυτό που κάνει το πρόγραμμα είναι να δημιουργεί ένα τυχαίο αριθμό καλώντας την Math.random() και μετά μέσω της κλάσης JOptionPane δίνει την ευκαιρία στο χρήστη να εισάγει ένα αριθμό. Αν ο αριθμός είναι μεγαλύτερος από αυτόν που δημιούργησε τυχαία το πρόγραμμα τότε βγάζει ένα μήνυμα το οποίο πληροφορεί το χρήστη ότι ο αριθμός ήταν μεγάλος. Το ίδιο συμβαίνει αν ο αριθμός είναι μικρότερος από το σωστό.

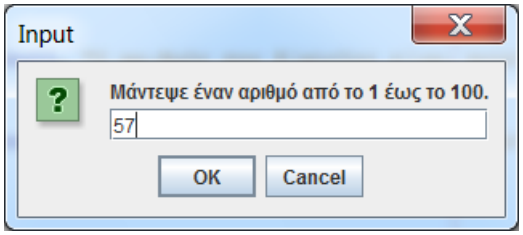
Η JOptionPane χρησιμοποιεί δύο μεθόδους στο πρόγραμμα: την showInputDialog (Object message) και την showMessageDialog (Component parentComponent, Object message). Η μέθοδος showInputDialog παίρνει ως παράμετρο το μήνυμα που θέλουμε να εμφανίζεται στο παράθυρο και επιστρέφει την τιμή που έχει εισάγει ο χρήστης στο κουτί κειμένου. Η μέθοδος showMessageDialog παίρνει παραμέτρους το συστατικό «πατέρα» (στο προκείμενο πρόγραμμα δεν υπάρχει οπότε βάζουμε null) και το μήνυμα που θέλουμε να πληροφορεί το χρήστη. Τέλος η showInputDialog είναι void μέθοδος έτσι δεν επιστρέφει κάποια τιμή.

Παρακάτω βλέπουμε τον κώδικα του προγράμματος:

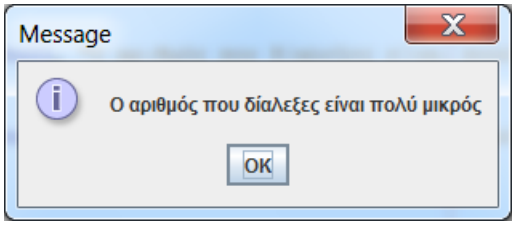
```
import javax.swing.*;

public class NumberGuesser {

      public static void main(String[] args) {

                  int guess = -1, count = 0;

            int num = (int)(Math.random()*100);

            do{

            guess = Integer.parseInt(JOptionPane.showInputDialog("Μάντεψε
            έναν       αριθμό από το 1 έως το 100."));
```

```
            if (guess>num)

                  JOptionPane.showMessageDialog(null,  "Ο  αριθμός  που
                  δίαλεξες είναι πολύ μεγάλος.");

            if (guess<num)

                   JOptionPane.showMessageDialog(null,  "Ο  αριθμός  που
                  δίαλεξες είναι πολύ μικρός");

            count++;

                   }while(num!=guess);

            JOptionPane.showMessageDialog(null,  "Συχγαρητήρια!  Μάντεψες
            το νούμερο " + num + " σε " + count + " προσπάθειες!!");

      }

}
```

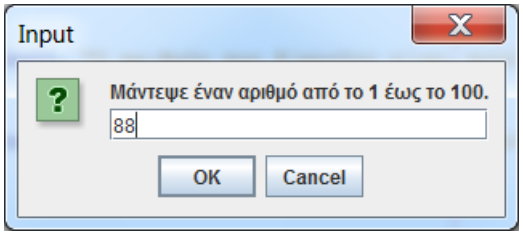
Τα παρακάτω σχήματα απεικονίζουν το Gameplay του παιχνιδιού “Number Guesser”:



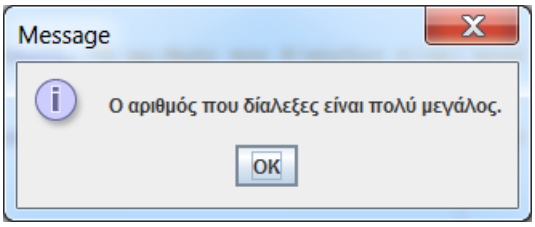
Σχήμα 3 "Υποθέτουμε το 57 ως το σωστό νούμερο"



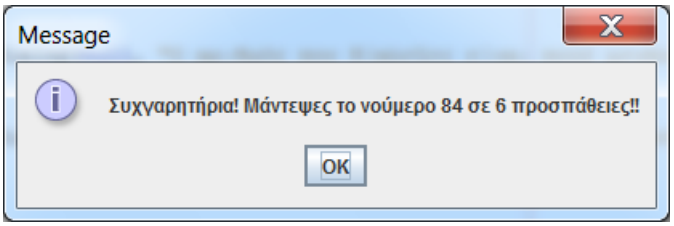
Σχήμα 4 "Το 57 είναι πολύ μικρό"



Σχήμα 5 Υποθέτουμε το 88 ως το σωστό νούμερο "



Σχήμα 6 " Το 88 είναι πολύ μεγάλο "



Σχήμα 7 "Το νούμερο 84 είναι το σωστό"

ΥΠΟΚΕΦΑΛΑΙΟ
1.2 ΤΟ ΜΟΝΤΕΛΟ ΥΠΟΔΟΧΕΑ ΡΙΖΙΚΟΥ ΠΛΑΙΣΙΟΥ

Το μοντέλο υποδοχέα ριζικού πλαισίου (Root Pane Container Model) εφαρμόζεται από ριζικές κλάσεις όπως: JDialog, JFrame, JWindow, JApplet και JInternalFrame. Προϋπόθεση ενός προγραμματιστή για να συντάξει κάποιο πρόγραμμα με τη Swing είναι να έχει γνώση του μοντέλου υποδοχέα ριζικού πλαισίου.

Στο AWT η τοποθέτηση των συστατικών πάνω σε ένα πλαίσιο (Frame) ή ένα Applet γίνεται με 2 τρόπους. Μπορούμε είτε να τοποθετήσουμε τα συστατικά πάνω στο πλαίσιο απευθείας είτε να δημιουργήσουμε ένα πανελ στο οποίο να προσθέσουμε όσα συστατικά θέλουμε και έπειτα να το τοποθετήσουμε στο πλαίσιο. Αντιθέτως η τοποθέτηση συστατικών σε ένα JFrame, JApplet, κ.λπ. είναι μια πιο σύνθετη διαδικασία. Ένα συνηθισμένο πρόβλημα είναι ότι αν προσπαθήσουμε να εισάγουμε κάποιο συστατικό απευθείας κατά τη διάρκεια της μεταγλώττισης, θα πάρουμε ένα μήνυμα σφάλματος. Αυτό συμβαίνει γιατί οι παραπάνω υποδοχείς χρησιμοποιούν ένα ειδικό μοντέλο τοποθέτησης.

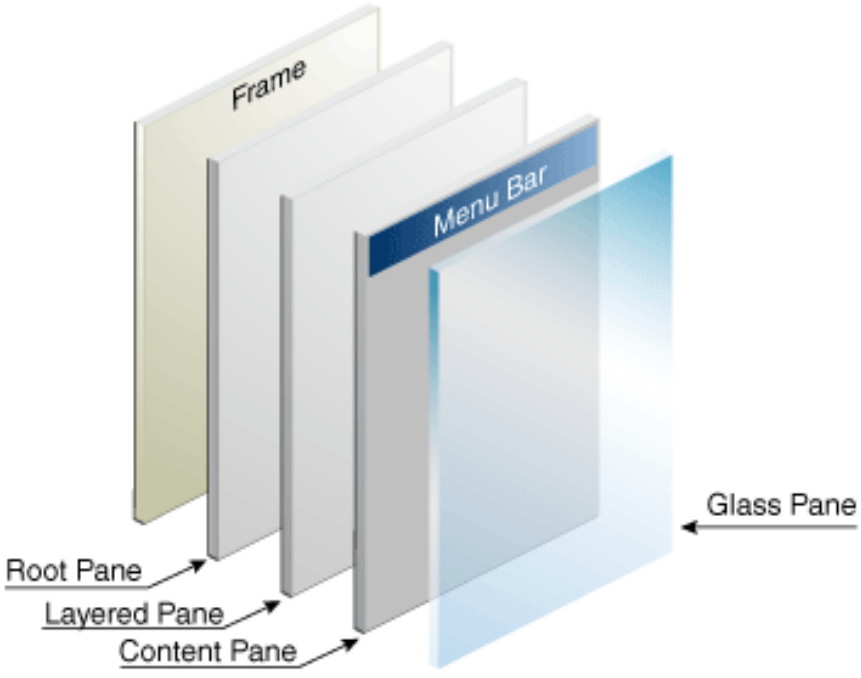
Ένα μοντέλο υποδοχέα ριζικού πλαισίου πρέπει να ακολουθεί κάποιες αρχές για να είναι σωστό. Κάθε εφαρμογή που κάνει χρήση των συστατικών της Swing θα πρέπει να έχει ένα τουλάχιστον ένα ριζικό υποδοχέα και να υλοποιεί τουλάχιστον μια ιεραρχία υποδοχής συστατικών. Για παράδειγμα, ο ριζικός υποδοχέας ενός Frame και ενός Applet είναι το JFrame και το JApplet αντίστοιχα. Επίσης αν σε ένα frame έχουμε ένα συστατικό διαλόγου τότε έχουμε δύο ριζικούς υποδοχείς: το JFrame και το JDialog. Επίσης κάθε ριζικός υποδοχέας διαθέτει επιμέρους χώρους. Αυτοί είναι:

- Ο ριζικός χώρος (Root Pane) ο οποίος είναι ο βασικός χώρος του ριζικού υποδοχέα και είναι αντικείμενο της κλάσης JRootPane.
- Ο διαστρωματωμένος χώρος (Layered Pane) στον οποίο μπορούμε να τοποθετήσουμε αντικείμενα και μάλιστα σε διάφορα επίπεδα. Επίσης αποτελεί αντικείμενο της κλάσης JLayeredPane.
- Ο χώρος περιεχομένων (Content Pane) ο οποίος βρίσκεται πάνω στο διαστρωματωμένο χώρο και είναι ο χώρος τοποθέτησης συστατικών της Swing. Ακόμα ο χώρος περιεχομένων είναι αντικείμενο της κλάσης

Container του πακέτου java.awt. Για να μπορέσουμε να αποκτήσουμε πρόσβαση στον Container και να μπορέσουμε να προσθέσουμε συστατικά σε αυτόν πρέπει να δηλώσουμε μια μεταβλητή τύπου Container και να καλέσουμε τη μέθοδο getContentPane().
Π.χ. Container cp = getContnentPane();

- Πέραν του χώρου περιεχομένων, πάνω στο διαστρωματωμένο χώρο μπορούμε να τοποθετήσουμε αν επιθυμούμε μια γραμμή μενού που είναι αντικείμενο της κλάσης JMenuBar. Η γραμμή μενού τοποθετείται στο πάνω μέρος, ενώ ο χώρος περιεχομένων στο κάτω μέρος του διαστρωματωμένου χώρου.
- Τέλος το υαλοπλαίσιο (Glass Pane) βρίσκεται πάνω από όλα τα προαναφερθέντα επίπεδα και μπορεί να χρησιμοποιηθεί για το χειρισμό συμβάντων που αφορούν αντικείμενα που είναι κάτω από αυτό ή για ομοιόμορφο χρωματισμό τέτοιων αντικειμένων.

Στο Σχήμα 8 μπορούμε να παρατηρήσουμε την ιεραρχία των επιπέδων που περιγράψαμε παραπάνω:



Σχήμα 8 "Η ιεραρχία επιπέδων του ριζικού υποδοχέα"

ΥΠΟΚΕΦΑΛΑΙΟ
1.3 ΤΟ ΠΛΑΙΣΙΟ - JFRAME

Το συστατικό JFrame αποτελεί το ριζικό υποδοχέα μιας παραθυρικής εφαρμογής και είναι αντίστοιχο του συστατικού Frame του AWT. Μόλις δημιουργήσουμε ένα αντικείμενο JFrame, στη συνέχεια μπορούμε να τοποθετήσουμε πάνω σε αυτό γραφικά συστατικά όπως ετικέτες, γραμμή μενού, κουμπιά, ράβδο κύλισης κ.λπ. Όπως αναφέραμε στην παράγραφο () το JFrame είναι βαρύ (heavyweight) χαρακτηριστικό που σημαίνει ότι κληρονομείται από το Frame.

Το Frame και το JFrame έχουν κάποιες σημαντικές διαφορές μεταξύ τους. Μερικές διαφορές είναι:

- Στο αντικείμενο Frame του AWT τα συστατικά (ετικέτες, κουμπιά, πλαίσια κειμένου κ.λπ.) μπορούν να τοποθετηθούν είτε απευθείας σε αυτό είτε μέσα σε ένα αντικείμενο πάνελ που στο τέλος τοποθετείται στο αντικείμενο Frame. Αντιθέτως, στο αντικείμενο JFrame δεν τοποθετούνται απευθείας αλλά καλούνε την μέθοδο getContentPane() για να μπορέσουν να εισάγουν συστατικά στο χώρο περιεχομένων.
- Στο αντικείμενο Frame όταν πατήσουμε το κουμπί κλεισίματος στο πλαίσιο με το ποντίκι, αυτό έχει ως αποτέλεσμα το κλείσιμο του παραθύρου και τον τερματισμό της εφαρμογής Java. Η διαφορά του με το JFrame είναι ότι το JFrame δεν τερματίζει αυτόματα την εφαρμογή και πρέπει να γράψουμε κώδικα για να τερματίζει την εφαρμογή όταν πατάμε το κουμπί κλεισίματος.

Η δημιουργία ενός πλαισίου σε Swing είναι πολύ απλή. Αρχικά πρέπει να ορίσουμε το αντικείμενο JFrame γράφοντας τον κώδικα:

```
JFrame frame = new JFrame("Εδώ γράφουμε τον τίτλο του πλαισίου");
```

Στην παράμετρο του JFrame βάζουμε τον τίτλο που θέλουμε να έχει το πλαίσιο. Αμέσως επόμενο βήμα είναι να ρυθμίσουμε το μέγεθος του πλαισίου:

```
frame.setSize(600, 500);
```

Έπειτα πρέπει να αποκτήσουμε πρόσβαση στο χώρο περιεχομένων με τη χρήση της μεθόδου getContentPane(). Στο χώρο περιεχομένων μπορούμε να καθορίσουμε το χρώμα φόντου του πλαισίου, το διαχειριστή διάταξης κ.λπ.

Στον κώδικα που ακολουθεί βλέπουμε τη δημιουργία του αντικειμένου χώρου περιεχομένων στο οποίο θα ορίσουμε άσπρο χρώμα για το φόντο του πλαισίου και το FlowLayout() διαχειριστή διάταξης. Τέλος στην τελευταία γραμμή κώδικα ρυθμίζουμε το πλαίσιο μας να είναι ορατό.

```
Container cont = getContentPane();

cont.setBackground(Color.white);

cont.setLayout(new FlowLayout());

frame.setVisible(true);
```

Όπως αναφέραμε ένα αντικείμενο JFrame δεν μπορεί να τερματίσει το πρόγραμμα από μόνο του, οπότε είμαστε αναγκασμένοι να γράψουμε κώδικα που να εκτελεί αυτή τη λειτουργία. Αυτό γίνεται απλά χρησιμοποιώντας την μέθοδο setDefaultCloseOperation(int operation) της JFrame στην οποία μπορούμε να θέσουμε μία λειτουργία όταν πατάμε το κουμπί κλεισίματος του πλαισίου. Οι διαθέσιμες λειτουργιές είναι:

- DO_NOTHING_ON_CLOSE: δεν κάνει καμία ενέργεια.
- HIDE_ON_CLOSE: κρύβει το πλαίσιο μας.
- DISPOSE_ON_CLOSE: κλείνει το πλαίσιο χωρίς να τερματίσει την εφαρμογή.
- EXIT_ON_CLOSE: κλείνει το πλαίσιο και τερματίζει την εφαρμογή.

Έτσι προσθέτουμε την παρακάτω γραμμή κώδικα για να τερματίσει και η εφαρμογή:

```
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

Τέλος αξίζει να σημειώσουμε ότι μπορούμε να δημιουργήσουμε κλάσεις που χρησιμοποιούν πλαίσια εάν ορίσουμε το JFrame ως «πατρική» κλάση. Αυτό γίνεται αν η κλάση μας κληρονομεί την κλάση JFrame.

```
class Example extends JFrame
```

ΥΠΟΚΕΦΑΛΑΙΟ

1.4 ΔΙΑΧΕΙΡΙΣΤΕΣ ΔΙΑΤΑΞΗΣ

Στην Java, την τοποθέτηση των διάφορων συστατικών σε μια εφαρμογή αναλαμβάνει ο διαχειριστής διάταξης (layout manager). Ο προεπιλεγμένος διαχειριστής διάταξης για το χώρο περιεχομένων ενός JFrame είναι ο BorderLayout, αλλά πέραν από αυτόν ο χρήστης μπορεί να ορίσει κάποιον άλλο διαχειριστή που να μπορεί να αντεπεξέλθει στις απαιτήσεις του. Υπάρχουν πέντε διαχειριστές διάταξης:

- Ο FlowLayout τοποθετεί τα συστατικά από αριστερά προς τα δεξιά και από επάνω προς τα κάτω.
- Ο BorderLayout τοποθετεί τα συστατικά σε πέντε θέσεις: NORTH (Βόρεια), SOUTH (Νότια), EAST (Ανατολικά), WEST (Δυτικά), CENTER (Κέντρο). Η θέση NORTH αντιστοιχεί στο επάνω μέρος του πλαισίου και οι άλλες θέσεις καθορίζονται σε σχέση με αυτή.
- Ο GrindLayout διαιρεί τον υποδοχέα σε ένα πλέγμα από γραμμές και στήλες στο οποίο το κάθε κελί έχει το ίδιο μέγεθος και η τοποθέτηση των συστατικών ξεκινάει από το πάνω-αριστερό κελί και προχωράει προς τα δεξιά.
- Ο GrindBagLayout είναι πιο πολύπλοκο από το GrindLayout. Το GrindBagLayout δημιουργεί ένα ορθογώνιο πλέγμα κελιών στο οποίο το μέγεθος του κελιού εξαρτάται από το συστατικό που θα προσθέσουμε, καθώς επίσης το κάθε συστατικό μπορεί να καταλαμβάνει ένα ή περισσότερα κελιά.
- Ο CardLayout είναι διαφορετικός από τους υπόλοιπους διαχειριστές επειδή κάθε συστατικό που τοποθετούμε στο πλαίσιο μας είναι μια νέα καρτέλα που τοποθετείται πάνω στις προϋπάρχουσες. Ο χρήστης μπορεί να μετακινηθεί εμπρός και πίσω στις καρτέλες αυτές.

ΥΠΟΚΕΦΑΛΑΙΟ

1.5 ΟΙ ΕΤΙΚΕΤΕΣ - JLABEL

Οι ετικέτες της Swing είναι παρόμοιες με αυτές της AWT και χρησιμοποιούνται για να εμφανίζουν μηνύματα. Ωστόσο τις κάνουν να ξεχωρίζουν

από αυτές της AWT επειδή προσφέρουν περισσότερες δυνατότητες. Στις ετικέτες της Swing πέραν από το κείμενο μπορούν να εμφανίζουν εικόνες, να έχουν περιθώρια, καθώς και να χρησιμοποιήσουν ετικέτες HTML για τη μορφοποίηση του περιεχομένου τους. Επίσης όπως ξέρουμε από προηγούμενη παράγραφο η κλάση JLabel είναι υποκλάση της JComponent.

Η JLabel έχει έξι κατασκευαστές. Αυτοί είναι:

- JLabel() – Ορίζει μία ετικέτα με κενό περιεχόμενο.
- JLabel(Icon εικόνα) – Ορίζει μία ετικέτα με εικόνα.
- JLabel(Icon εικόνα, int ευθυγράμμιση) - Ορίζει μία ετικέτα με εικόνα και ευθυγράμμιση εικόνας σε σχέση με την ετικέτα (η ευθυγράμμιση μπορεί να πάρει τίμες: LEFT, CENTER, RIGHT, LEADING, EAST, WEST, κ.λπ).
- JLabel(String κείμενο) – Ορίζει μία ετικέτα με κείμενο.
- JLabel(String κείμενο, Icon εικόνα, int ευθυγράμμιση) - Ορίζει μία ετικέτα με κείμενο, εικόνα και ευθυγράμμιση εικόνας-κειμένου σε σχέση με την ετικέτα.
- JLabel(String κείμενο, int ευθυγράμμιση) - Ορίζει μία ετικέτα με εικόνα και ευθυγράμμιση κειμένου σε σχέση με την ετικέτα.

Η δημιουργία ενός αντικειμένου Icon και ο ορισμός της εικόνας που θέλουμε γίνεται ως εξής:

```
Icon img = new ImageIcon("image.jpg");
```

Η κλάση JLabel έχει αρκετές μεθόδους που μας διευκολύνουν στην σχεδίαση των ετικετών. Μερικές χρήσιμες μέθοδοι είναι:

- setDisabledIcon(Icon εικόνα) – Απενεργοποιεί μία εικόνα που βρίσκεται στην ετικέτα.
- setHorizontalAlignment(int στοίχιση) – Ορίζει τη στοίχιση του περιεχομένου της ετικέτας κατά μήκος του X άξονα. Οι τιμές που μπορεί να πάρει η οριζόντια στοίχιση είναι LEFT, CENTER, RIGHT, LEADING και TRAILING.
- setHorizontalTextPosition(int θέση) - Ορίζει την οριζόντια θέση του κειμένου σε σχέση με την εικόνα μιας ετικέτας.
- setIcon(Icon εικόνα) – Ορίζει την εικόνα της ετικέτας.
- setText(String κείμενο) - Ορίζει το κείμενο της ετικέτας.

- setVerticalAlignment(int στοίχιση) - Ορίζει τη στοίχιση του περιεχομένου της ετικέτας κατά μήκος του Y άξονα. Οι τιμές που μπορεί να πάρει η κατακόρυφη στοίχιση είναι TOP, CENTER και BOTTOM.
- setVerticalTextPosition(int textPosition) - Ορίζει την κατακόρυφη θέση του κειμένου σε σχέση με την εικόνα μιας ετικέτας.

Επιπλέον η JLabel κληρονομεί ένα μεγάλο αριθμό μεθόδων από την κύρια κλάση JComponent και την Component. Οι πιο σημαντικές για την JLabel είναι:

- setBackground(Color χρώμα) – Ορίζει το χρώμα του φόντου της ετικέτας.
- setBounds(int x, int y, int μήκος, int ύψος) – Το x και το y ορίζει το που θα τοποθετηθεί στο πλαίσιο η ετικέτα και το μήκος και ύψος ορίζουν το μέγεθος της ετικέτας.
- setFont(Font γραμματοσειρά) – Ορίζει τη γραμματοσειρά της ετικέτας.
- setForeground(Color χρώμα) - Ορίζει το χρώμα του πρώτου πλάνου της ετικέτας.

Αξίζει να πούμε ότι ο κατασκευαστής της Font ορίζεται ως:

Font(String όνομα_γραμματοσειράς, int ύφος_κειμένου, int μέγεθος)

όπου το όνομα γραμματοσειράς μπορεί να είναι “Arial”, “Calibri”, κ.λπ., το ύφος κειμένου παίρνει τιμές όπως PLAIN, BOLD και ITALIC, και το μέγεθος κειμένου παίρνει αριθμητικές τιμές.

ΥΠΟΚΕΦΑΛΑΙΟ
1.6 ΤΑ ΚΟΥΜΠΙΑ - JBUTTON

Ένα κουμπί στη Swing είναι αντικείμενο της κλάσης JButton, η οποία κληρονομεί από την κλάση AbstractButton. Σε ένα κουμπί της Swing μπορούμε να τοποθετήσουμε κείμενο, εικόνα ή και τα δύο. Ακόμα μπορούμε να ορίζουμε όσα κουμπιά επιθυμούμε στην εφαρμογή μας. Η λειτουργία του κουμπιού γίνεται ως εξής: όταν χρήστης πατάει στο κουμπί με το δείκτη του ποντικού τότε δημιουργείται ένα γεγονός (event) στο οποίο καθορίζουμε τι θέλουμε να γίνεται όταν πατάμε αυτό το κουμπί. Για να μπορεί να υπάρξη χειρισμός του κουμπιού

πρέπει να δημιουργήσουμε ένα ActionLister αντικείμενο και να προσθέσουμε το κουμπί σε αυτό μέσω της addActionListener μεθόδου.

Οι βασικοί κατασκευαστές του JButton είναι:

- JButton() – Δημιουργεί ένα κουμπί χωρίς περιεχόμενο.
- JButton(Action ενέργεια) – Δημιουργεί ένα κουμπί όπου οι ιδιότητές του ορίζονται από μια ενέργεια.
- JButton(Icon εικόνα) – Δημιουργεί ένα κουμπί με μία εικόνα.
- JButton(String κείμενο) - Δημιουργεί ένα κουμπί με κείμενο.
- JButton(String κείμενο, Icon εικόνα) - Δημιουργεί ένα κουμπί με κείμενο και εικόνα.

Κάποιες σημαντικές μέθοδοι της JButton είναι παρόμοιες με αυτές της JLabel που αναφέραμε στην προηγούμενη παράγραφο. Αυτές είναι:

- setDisabledIcon(Icon εικόνα)
- setFont(Font γραμματοσειρά)
- setHorizontalAlignment(int στοίχιση)
- setHorizontalTextPosition(int θέση)
- setIcon(Icon εικόνα)
- setText(String κείμενο)
- setVerticalAlignment(int στοίχιση)
- setVerticalTextPosition(int textPosition)

Αυτές που χρησιμοποιούνται μόνο για τα JButtons είναι:

- addActionListener(ActionListener al) – Προσθέτει ένα ActionListener σε ένα κουμπί.
- setAction(Action a) – Ορίζει την ενέργεια για την ActionEvent πηγή.
- setPressedIcon(Icon εικόνα) – Εμφανίζει την εικόνα όταν πατάμε το κουμπί.
- setPressedIcon(Icon pressedIcon) - Εμφανίζει την εικόνα όταν ο δείκτης του ποντικιού κινείται πάνω στο κουμπί.
- removeActionListener(ActionListener al) – Αφαιρεί ένα ActionListener από ένα κουμπί.

ΥΠΟΚΕΦΑΛΑΙΟ
1.7 Η ΚΛΑΣΗ JFILECHOOSER

Μια ακόμα χρήσιμη κλάση της Swing είναι η JFileChooser. Πολλές φορές ο χρήστης έχει την ανάγκη σε μια εφαρμογή ή ένα παιχνίδι να ανοίξει ή να αποθηκεύσει κάποιο αρχείο. Αυτό μπορεί να επιτευχθεί με τη χρήση της κλάσης JFileChooser η οποία δίνει τη δυνατότητα στον χρήστη να επιλέξει ένα αρχείο για άνοιγμα ή αποθήκευση μέσα από ένα παράθυρο διαλόγου. Ένα άλλο στοιχείο που μπορούμε να προσθέσουμε είναι η FileFilter. Η κλάση FileFilter εμφανίζει μόνο ένα ή περισσότερους τύπους αρχείων, που ο προγραμματιστής έχει ορίσει, στο πλαίσιο διαλόγου. Έτσι με αυτόν τον τρόπο διευκολύνει το χρήστη στην επιλογή και αποφεύγεται κάποιο πιθανό σφάλμα. Τέλος μπορούμε να θέσουμε τον κατάλογο (directory) που θα ανοίγει το πλαίσιο διαλόγου όταν επιλέγουμε ένα αρχείο.

Οι τρεις σημαντικοί κατασκευαστές της κλάσης JFileChooser είναι:

- JFileChooser() – Δημιουργεί ένα JFileChooser με προεπιλεγμένο κατάλογο.
- JFileChooser(File τρέχων_κατάλογος) - Δημιουργεί ένα JFileChooser το οποίο παίρνει ως τρέχοντα κατάλογο το μονοπάτι του αρχείου.
- JFileChooser(String μονοπάτι_τρέχοντος_καταλόγου) - Δημιουργεί ένα JFileChooser το οποίο παίρνει ως τρέχοντα κατάλογο το μονοπάτι που του ορίζουμε.

Μερικές βασικές μέθοδοι που πρέπει να γνωρίζουμε είναι:

- setCurrentDirectory(File αρχείο) – Ορίζει το μονοπάτι του αρχείου ως τρέχοντα κατάλογο στο πλαίσιο διαλόγου.
- setDialogTitle(String τίτλος) – Ορίζει ένα τίτλο στο πλαίσιο διαλόγου.
- setFileFilter(FileFilter φίλτρο) - Ορίζει ένα φίλτρο στο πλαίσιο διαλόγου.
- showOpenDialog(Component γονιός) – Εμφανίζει ένα πλαίσιο διαλόγου για άνοιγμα αρχείου. Αν δεν θέλουμε να μπει «πατρικό» πλαίσιο, χρησιμοποιούμε το όρισμα null.
- showSaveDialog(Component γονιός) - Εμφανίζει ένα πλαίσιο διαλόγου για αποθήκευση αρχείου. Αν δεν θέλουμε να μπει «πατρικό» πλαίσιο, χρησιμοποιούμε το όρισμα null.

ΥΠΟΚΕΦΑΛΑΙΟ
1.8 ΤΟ ΜΕΝΟΥ ΠΛΑΙΣΙΟΥ

Η ύπαρξη του μενού σε ένα πρόγραμμα ή ένα παιχνίδι παίζει σημαντικό ρόλο στην λειτουργικότητά του. Η προσθήκη μίας μενού μπάρας στο πλαίσιο μας δίνει τη δυνατότητα να εισάγουμε κατηγορία και σε κάθε κατηγορία να ορίσουμε διάφορα κουμπιά όπου το καθένα θα εκτελεί μία λειτουργία στο πρόγραμμα όπως αποθήκευση, κλείσιμο, επανεκκίνηση, κ.λπ. Ένα μενού αποτελείται από τρία βασικά μέρη:

1. τη γραμμή μενού που λειτουργεί ως υποδοχέας για τα μενού
2. τα ίδια τα μενού που ανοίγουν και κλείνουν
3. τα στοιχεία του κάθε μενού (menuItems) που εμφανίζονται όταν το μενού ανοίγει και αποκρύπτονται όταν κλείνει

Οι κλάσεις που παρέχει η Swing για τη δημιουργία μενού στα πλαίσια είναι η JMenu, η JMenuBar και η JMenuItem. Η κλάση JMenuBar κληρονομεί από τη JComponent και προσφέρει μεθόδους για το χειρισμό της γραμμής μενού. Η κλάση JMenuItem προσφέρει μεθόδους για το χειρισμό των στοιχείων που υπάρχουν στα μενού. Τέλος η κλάση JMenu προσφέρει μεθόδους για τον χειρισμό των μενού που περιέχουν στοιχεία μενού. Επιπλέον υπάρχουν δύο ακόμα χρήσιμες κλάσεις: η JCheckBoxMenuItem και JRadioButtonMenuItem τις οποίες χρησιμοποιούμε όταν θέλουμε τα στοιχεία του μενού να είναι πλαίσια ελέγχου ή ραδιόπληκτρα.

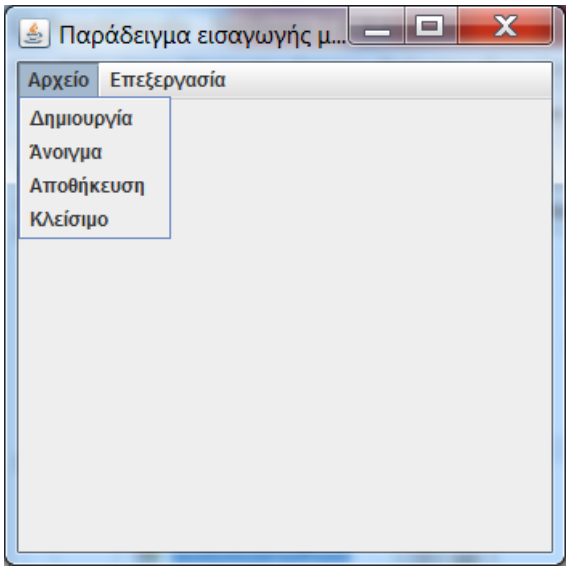
Παρακάτω ακολουθεί ο κώδικας ενός απλού παραδείγματος εισαγωγής μενού σε ένα πλαίσιο:

```
import java.awt.*;      //Ορισμός βιβλιοθηκών awt, swing και util
import java.awt.event.*;
import java.util.StringTokenizer;
import javax.swing.*;

public class MenuExample extends JFrame implements ActionListener{

    Container cont;
```

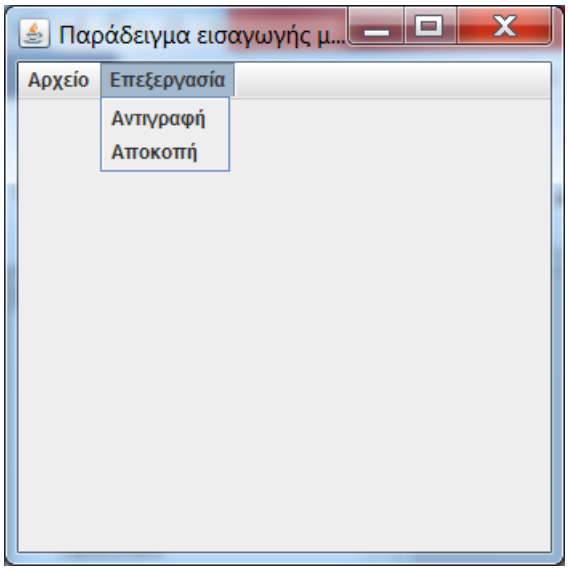
```
    JMenu file, process;    //Ορισμός των 2 μενού που θα έχει η ράβδος
    JMenuBar menuBar;

    public MenuExample(){
        super("Παράδειγμα εισαγωγής μενού");   //Δημιουργία του πλαισίου
        this.setVisible(true);
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        this.setSize(350,350);
        cont = getContentPane();
        cont.setLayout(null);

        menuBar =  new JMenuBar();
        file= new JMenu("Αρχείο");
        //Προσθήκη των στοιχείων στο 1ο μενού
        file.add(createMenuItem("Αρχείο", "Δημιουργία", this));
        file.add(createMenuItem("Αρχείο", "Άνοιγμα", this));
        file.add(createMenuItem("Αρχείο", "Αποθήκευση", this));
        file.add(createMenuItem("Αρχείο", "Κλείσιμο", this));
        menuBar.add(file);  //Προσθήκη του 1° μενού στη μπάρα μας

        process= new JMenu("Επεξεργασία");
        //Προσθήκη των στοιχείων στο 2ο μενού
        process.add(createMenuItem("Επεξεργασία", "Αντιγραφή", this));
        process.add(createMenuItem("Επεξεργασία", "Αποκοπή", this));
        menuBar.add(process); //Προσθήκη του 2ο μενού στη μπάρα μας

        this.setJMenuBar(menuBar); //Προσθήκη της μπάρας στο πλαίσιο
    }

    public JMenuItem createMenuItem(String menuText, String buttonText,
ActionListener listener){
//Δημιουργεί τα στοιχεία του κάθε μενού και προσθέτει στο καθένα ένα
ActionListener.
        JMenuItem menuItem = new JMenuItem(buttonText);
```

```
        menuItem.addActionListener(listener);
        menuItem.setActionCommand(menuText + "|" + buttonText);
        return menuItem;
    }
    public void actionPerformed(ActionEvent ae) {
        //Εμφανίζει στην οθόνη το μενού και το στοιχείο του που διαλέξαμε
        System.out.println("Action Performed: " + ae.getActionCommand());
       //Χωρίζει την ενέργεια μας σε 2 strings
        StringTokenizer token = new
StringTokenizer(ae.getActionCommand(), "|");
        String menu = token.nextToken();
        String command = token.nextToken();
    }
    public static void main(String args[]){
        new MenuExample();
    }
}
```

Αρχικά το πρόγραμμα δημιουργεί ένα πλαίσιο με 350 pixels μήκος και πλάτος και τίτλο «Παράδειγμα εισαγωγής μενού». Εν συνεχεία δημιουργούμε το πρώτο μενού που είναι το «Αρχείο» και μέσω της μεθόδου createMenuItem προσθέτουμε τα στοιχεία: Δημιουργία, Άνοιγμα, Αποθήκευση και Κλείσιμο σε αυτό (Σχήμα 9). Ουσιαστικά η μέθοδος createMenuItem δημιουργεί ένα στοιχείο και προσθέτει έναν ακροατή actionListener σε αυτό. Αφού τελειώσει η προσθήκη των στοιχείων, τοποθετείται το πρώτο μενού στη μπάρα. Το ίδιο γίνεται και για το δεύτερο μενού που ονομάζεται «Επεξεργασία» και περιέχει τα στοιχεία: Αντιγραφή και Αποκοπή (Σχήμα 10). Τέλος κάθε φορά που πατάμε ένα στοιχείου ενός από τα μενού ενεργοποιείται η actionPerformed, η οποία κάνεις τις εξής ενέργειες: εμφανίζει το μενού και το στοιχείο που πατήσαμε στην οθόνη και χωρίζει το αλφαριθμητικό της ενέργειας που εκτελέστηκε σε δυο αλφαριθμητικά ορίζοντας ως σύμβολο χωρισμού το «|». Το πρώτο περιέχει το μενού και το δεύτερο το στοιχείο που πατήσαμε. Χάρη στην actionPerformed μπορούμε να εντοπίζουμε ποιο στοιχείο πάτησε ο χρήστης και να ορίσουμε με κώδικα την λειτουργία του κάθε στοιχείου.



Σχήμα 9 "Τα στοιχεία του μενού Αρχείο"



Σχήμα 10 "Τα στοιχεία του μενού Επεξεργασία "

ΥΠΟΚΕΦΑΛΑΙΟ
1.9 Ο ΕΠΙΛΟΓΕΑΣ ΧΡΩΜΑΤΩΝ - JCOLORCHOOSER

Η JColorChooser είναι ένα χρήσιμο συστατικό, που παρέχει μόνο η Swing, το οποίο μας δίνει τη δυνατότητα να επιλέξουμε πάρα πολλά χρώματα και να τα χρησιμοποιήσουμε στην εφαρμογή μας. Η επιλογή χρώματος μπορεί να γίνει με πέντε διαφορετικούς τρόπους:

- Στην πρώτη καρτέλα του πλαισίου επιλογέα χρωμάτων εμφανίζεται μια μεγάλη γκάμα χρωμάτων (Swatches) στην οποία το κάθε χρώμα βρίσκεται μέσα σε ένα μικρό τετραγωνάκι. Ο χρήστης μπορεί να επιλέξει με το ποντίκι του το χρώμα που επιθυμεί να χρησιμοποιήσει (Σχήμα 13).
- Η δεύτερη καρτέλα ονομάζεται HSV (Hue, Saturation, Value) και δίνει τη δυνατότητα στο χρήστη να αναμείξει τέσσερις τιμές για να φτιάξει ένα χρώμα. Οι τιμές αυτές είναι η απόχρωση, η διαπότιση, η φωτοσκίαση και η διαφάνεια. Η τιμές τις απόχρωσης κυμαίνονται από 0 μέχρι 360, ενώ οι υπόλοιπες από 0 μέχρι 100.
- Η τρίτη καρτέλα ονομάζεται HSL (Hue, Saturation, Lightness) και είναι σχεδόν ίδια με την HSV με τη διαφορά ότι αντί για φωτοσκίαση (Value) έχουμε απαλότητα (Lightness) για τιμή. Η απαλότητα παίρνει τιμές από 0 μέχρι 100.

- Η τετάρτη καρτέλα είναι η RGB (Red, Green, Blue) όπου ο χρήστης πρέπει να αναμίξει τις τιμές των χρωμάτων κόκκινου, πράσινου και μπλε, οι οποίες κυμαίνονται από 0 μέχρι 255.
- Η πέμπτη καρτέλα είναι η CMYK (Cyan, Magenta, Yellow, Black) όπου ο χρήστης πρέπει να αναμίξει τις τιμές των χρωμάτων κυανού, ζωηρού κόκκινου, κίτρινου και μαύρου, οι οποίες κυμαίνονται από 0 μέχρι 255.

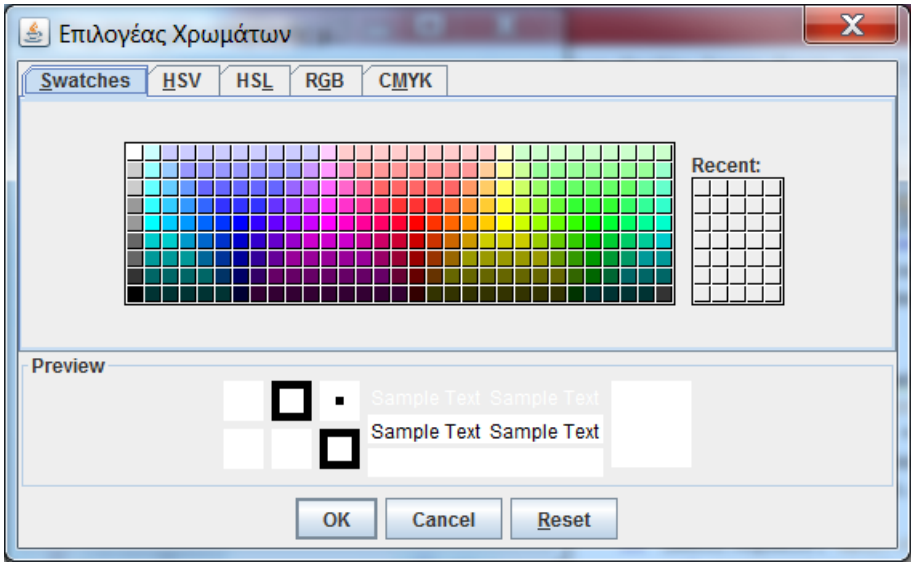
Οι δύο βασικοί κατασκευαστές της JColorChooser είναι:

- JColorChooser() – Δημιουργεί ένα αντικείμενο επιλογέα χρωμάτων έχοντας ως προεπιλεγμένο χρώμα το άσπρο.
- JColorChooser(Color αρχικό_χρώμα) - Δημιουργεί ένα αντικείμενο επιλογέα χρωμάτων με το αρχικό χρώμα που ορίζει ο χρήστης.

Η μέθοδος της κλάσης JColorChooser που χρησιμοποιούμε για να εμφανίσουμε το παράθυρο διαλόγου επιλογής χρώματος είναι:

showDialog(Component συστατικό, String τίτλος, Color αρχικό_χρώμα)

Η showDialog παίρνει ως όρισμα ένα συστατικό, τον τίτλο του πλαισίου διαλόγου και το χρώμα που θα είναι προεπιλεγμένο όταν ανοίξει το πλαίσιο. Αν ο χρήστης πατήσει το κουμπί OK τότε θα κλείσει το πλαίσιο και θα επιστρέψει το χρώμα που επέλεξε. Αν πατήσει Cancel τότε θα κλείσει το πλαίσιο και θα επιστρέψει null. Τέλος αν πατήσει Reset θα επαναφέρει το τρέχον χρώμα στο προεπιλεγμένο (Σχήμα 11).



Σχήμα 11 "Το πλαίσιο διαλόγου επιλογής χρώματος"

ΕΠΙΛΟΓΟΣ

Στο κεφάλαιο αυτό αναφερθήκαμε στον τρόπο με το οποίο μπορούμε να φτιάξουμε ένα πλαίσιο JFrame και αναλύσαμε τα απαραίτητα συστατικά της JFrame, που χρειαζόμαστε στη δημιουργία του κορμού ενός παιχνιδιού ή γενικά ενός προγράμματος γραφικών. Στο επόμενο κεφάλαιο θα περιγράψουμε τη λειτουργία των νημάτων στη Java και τη χρήση τους στα παιχνίδια και animations.

ΚΕΦΑΛΑΙΟ 2
ΤΑ ΝΗΜΑΤΑ ΣΤΗ JAVA

ΕΙΣΑΓΩΓΗ

Παλιότερα οι υπολογιστές δεν είχαν τη δυνατότητα ταυτόχρονης εκτέλεσης δύο ή περισσότερων ενεργειών, με αποτέλεσμα να μπορούν να εκτελούν μόνο μία ενέργεια κάθε χρονική στιγμή. Αυτό συνέβαινε επειδή η εκτέλεση των προγραμμάτων ήταν σειριακή και κάθε φορά το εκτελούμενο πρόγραμμα είχε τον απόλυτο έλεγχο του υπολογιστή. Όταν πρωτοεμφανίστηκαν τα λειτουργικά συστήματα καταμερισμού χρόνου (multisharing) έδωσαν λύση στο πρόβλημα της σειριακής εκτέλεσης και κατόρθωσαν να κάνουν έναν υπολογιστή να μοιράζει το χρόνο λειτουργίας του σε πολλούς χρήστες και να εκτελεί ταυτόχρονα τα διάφορα προγράμματά τους. Η διαδικασία αυτή γινόταν ως εξής: το λειτουργικό σύστημα διαιρούσε το χρόνο του συστήματος και παραχωρούσε ένα χρονικό παράθυρο στο πρόγραμμα κάθε χρήστη για την εκτέλεσή του. Το επόμενο τεχνολογικό βήμα που έγινε ήταν η ανάπτυξη υπολογιστών και λειτουργικών συστημάτων τα οποία υποστήριζαν την πολυδιεργασία (multitasking). Η πολυδιεργασία επέτρεπε στον χρήστη να τρέχει ταυτόχρονα πολλά προγράμματα στον ίδιο υπολογιστή. Αυτό ήταν εφικτό διότι το κάθε πρόγραμμα ακολουθούσε τη δικιά του πορεία εκτέλεσης ανεξάρτητα από τα άλλα προγράμματα που τρέχαν ταυτόχρονα με αυτό.

Ωστόσο υπάρχουν φορές όπου κάποια προγράμματα χρειάζονται να εκτελέσουν ταυτόχρονα πολλές εργασίες. Αυτή τη δουλειά κάνουν τα νήματα τα οποία επιτρέπουν σε κάθε πρόγραμμα να μπορεί να τρέχει ταυτόχρονα περισσότερες από μία διεργασίες. Ο αριθμός των νημάτων που τρέχουν σε κάθε χρονική στιγμή εξαρτάται από αριθμό των διαθέσιμων κεντρικών μονάδων επεξεργασίας (CPU). Κάποια νήματα μπορούν στην πραγματικότητα να τρέχουν παράλληλα σε διαφορετικές CPU το καθένα. Ένα πρόγραμμα όπου κάνει χρήση των νημάτων συχνά είναι το πρόγραμμα ανάγνωσης ιστοσελίδων το οποίο μπορεί να κατεβάζει ένα αρχείο, να εμφανίζει μια ιστοσελίδα στην οθόνη και να τυπώνει ένα αρχείο ταυτόχρονα.

ΥΠΟΚΕΦΑΛΑΙΟ
2.1 Η ΧΡΗΣΗ ΤΩΝ ΝΗΜΑΤΩΝ ΣΤΗ JAVA

Τα νήματα μπορούν να αποτελέσουν πολύτιμο εργαλείο για το χειρισμό πολύπλοκων εφαρμογών. Η Java έχει σχεδιαστεί να χρησιμοποιεί νήματα και είναι ευκολότερο να δουλεύουμε με νήματα στη Java σε σύγκριση με άλλες γλώσσες προγραμματισμού. Κάθε applet ή εφαρμογή της Java έχει τουλάχιστον ένα νήμα. Επίσης τα νήματα παίζουν σημαντικό ρόλο στην ανάπτυξη των παιχνιδιών και κινούμενων σχεδίων (animation) τα οποία χρειάζονται να εκτελούν πολλές ενέργειες ταυτόχρονα.

Για να δημιουργήσουμε και να τρέξουμε ένα νέο νήμα, απλά ορίζουμε μία μέθοδο start():

```
public void start() {
        Thread myTread = new Thread();
        myThread.start();
}
```

Ο παραπάνω κώδικας δεν κάνει τίποτα χρήσιμο διότι δεν έχει ανατεθεί μία εργασία στο νήμα. Η κλήση της μεθόδου start() δεν συνεπάγεται και άμεση εκτέλεση του νήματος. Απλά το νήμα αποκτά δικαίωμα να τρέξει σε χρόνο που θα του παραχωρήσει η κεντρική μονάδα επεξεργασίας. Εάν θέλουμε να αναθέσουμε μία εργασία σε ένα νήμα απλά ορίζουμε τη μέθοδο run() και προσθέτουμε κώδικα για να κάνει κάτι.

Υπάρχουν τρεις τρόποι που μπορούμε να κάνουμε αυτό το πράγμα:

1. **Απευθύνοντας στην κλάση Thread.**

   Ένας γρήγορος τρόπος για να αναθέσουμε μία εργασία σε ένα νήμα είναι να γράψουμε μία κλάση η οποία να απευθύνεται στην κλάση Thread. Επειδή η κλάση Thread εφαρμόζει τη διασύνδεση runnable, εμπεριέχει τη μέθοδο run(), την οποία όμως ο χρήστης πρέπει να παρακάμψει (override) ώστε το νήμα να εκτελεί αυτά που θέλει:

```
public class MyThread extends Thread {
   public void run() {
        System.out.println("Κάνε μία εργασία");
   }
}
```

Έπειτα δημιουργούμε και ξεκινάμε το νήμα απλά ορίζοντας μια μέθοδο start() όπως είπαμε παραπάνω.

2. Εφαρμόζοντας την διασύνδεση Runnable

Ο προηγούμενος τρόπος είναι σχετικά εύκολος, αλλά τις περισσότερες φορές δεν θα θέλουμε να γράψουμε μία νέα κλάση για να ξεκινήσουμε ένα νήμα. Για παράδειγμα μπορεί να θέλουμε μία κλάση να απευθύνεται σε μία άλλη κλάση και να τρέχει σαν νήμα. Σε αυτήν την περίπτωση εφαρμόζουμε την διασύνδεση Runnable:

```
public MyClass extends SomeOtherClass implements Runnable {
   public MyClass() {
         Thread thread = new Thread();
         Thread.start();
   }
   public void run() {
         System.out.println("Κάνε μία εργασία");
   }
}
```

Σε αυτό το παράδειγμα, ο κατασκευαστής MyClass δημιουργεί και ξεκινάει ένα νέο νήμα. Η κλάση Thread παίρνει ένα Runnable αντικείμενο ως παράμετρο μέσα στον κατασκευαστή της, και το Runnable αντικείμενο εκτελείται όταν το νήμα ξεκινάει.

3. Χρησιμοποιώντας μία ανώνυμη εσωτερική κλάση

Μερικές φορές χρειάζεται να δημιουργήσουμε ένα καινούργιο νήμα χωρίς να θέλουμε να φτιάξουμε μία νέα κλάση ή δεν είναι βολικό να εφαρμόσουμε την Runnable διασύνδεση. Σε αυτή την περίπτωση μπορούμε να χρησιμοποιήσουμε μια ανώνυμη εσωτερική κλάση η οποία θα ξεκινάει ένα νέο νήμα:

```
new Thread() {
   public void run() {
         System.out.println("Κάνε μία εργασία");
   }
}.start();
```

Αυτό το παράδειγμα είναι αρκετά απλό, αλλά μπορεί εύκολα να μην διαβάζεται αν ο κώδικας μας στη μέθοδο run() είναι πολύ μακρύς.

Αξίζει να σημειώσουμε ότι μερικές φορές μπορεί να χρειαζόμαστε διακόψουμε για μικρό χρονικό διάστημα ένα νήμα. Για να το επιτύχουμε αυτό χρησιμοποιούμε την στατική μέθοδο sleep():

```
Thread.sleep(1000);
```

Αυτή η γραμμή κώδικα αναγκάζει το νήμα να «κοιμηθεί» για 1000 χιλιοστά του δευτερολέπτου ή οποιοδήποτε άλλο χρονικό διάστημα θέσουμε. Το σταματημένο νήμα δεν καταναλώνει καθόλου χρόνο στη CPU. Η μέθοδος sleep() ενδέχεται να δημιουργήσει την εξαίρεση InterruptedException και για αυτό να τοποθετείται σε μπλοκ try/catch.

Εάν θέλουμε το τρέχον νήμα να περιμένει μέχρι ένα άλλο νήμα τελειώσει, χρησιμοποιούμε την μέθοδο join():

```
myThread.join();
```

Αυτή είναι χρήσιμη όταν ένας παίκτης βγαίνει από το παιχνίδι, τότε θα πρέπει να βεβαιωθούμε ότι όλα τα νήματα έχουν τελειώσει πριν καθαρίσουμε τα αντικείμενα.

Κάθε νήμα μπορεί να βρεθεί σε διάφορες καταστάσεις:

- Η αρχική είναι η κατάσταση γέννησης (born state) στην οποία βρίσκεται όταν δημιουργείται.
- Μόλις κληθεί η μέθοδος start() του νήματος τότε μεταβαίνει σε κατάσταση ετοιμότητας (ready state).
- Έπειτα ο χρονοπρογραμματιστής το οδηγεί σε κατάσταση εκτέλεσης (running state) που είναι ο προορισμός του κάθε νήματος.
- Όταν τερματίσει η μέθοδος run() ενός νήματος τότε αυτό τίθεται σε κατάσταση τερματισμού (dead state) από την οποία δεν μπορεί να επιστρέψει σε ετοιμότητα.
- Επίσης ένα νήμα σε κατάσταση εκτέλεσης μπορεί να βρεθεί σε διάφορες κατατάσσεις αναμονής από τις οποίες μπορεί να επανέλθει και πάλι σε κατάσταση ετοιμότητας και μετά σε κατάσταση εκτέλεσης:
    - Την κατάσταση “sleeping” (με κλήση της μέθοδου sleep)
    - Την κατάσταση “waiting” (με κλήση της μεθόδου wait)
    - Την κατάσταση “blocked”

ΥΠΟΚΕΦΑΛΑΙΟ
2.2 ΣΥΓΧΟΝΙΣΜΟΣ

Κάθε νήμα εκτελείται ανεξάρτητα το ένα από το άλλο και δεν χρειάζεται να είναι ενήμερο για τις εργασίες των άλλων νημάτων. Όμως υπάρχουν μερικές περιπτώσεις όπου τα νήματα χρειάζεται να μοιράζονται δεδομένα, έτσι ώστε να μην προσπαθούν πολλαπλά νήματα να αποκτήσουν πρόσβαση στο ίδιο αντικείμενο ή στην ίδια μεταβλητή ταυτόχρονα. Το πρόβλημα που παρουσιάζεται όταν διαφορετικά αντικείμενα μιας κλάσης διαχειρίζονται στατικές μεταβλητές ή στατικές μεθόδους ονομάζεται πρόβλημα συγχρονισμού.

Για να αποφευχθεί το πρόβλημα της πολλαπλής πρόσβασης σε δεδομένα, η Java επιτρέπει το κλείδωμα των αντικειμένων και των δεδομένων με τη χρήση της δεσμευμένης λέξης synchronized. Ο συγχρονισμός περιορίζει την προσπέλαση μιας μεθόδου σε μόνο ένα νήμα κάθε φορά. Για να ορίζουμε μία μέθοδο ως synchronized γράφουμε:

```
public synchronized void myMethod()  { }
```

Εάν δεν θέλουμε να δηλώσουμε ολόκληρη τη μέθοδο synchronized, τότε μπορούμε να δηλώσουμε ένα τμήμα κώδικα μέσα στη μέθοδο που επιθυμούμε ως synchronized:

```
Public int myMethod() {
      Synchronized(this){
      }
}
```

Για να προσπελάσει ένα νήμα μια μέθοδο που έχει δηλωθεί ως synchronized, πρέπει να περιμένει να τελειώσει το τρέχον νήμα. Όταν ένα νήμα περιμένει ένα άλλο να τελειώσει και να ελευθερωθεί το κλείδωμα που υπάρχει σε μία μέθοδο ή σε ένα αντικείμενο, τότε είναι μπλοκαρισμένο.

Ας πούμε ότι θέλουμε να φτιάξουμε ένα παιχνίδι με λαβύρινθο. Οποιαδήποτε νήμα μπορεί να θέσει τη θέση του παίκτη και κάθε νήμα μπορεί να ελέγξει αν ο παίκτης βρίσκεται στην έξοδο του λαβύρινθου. Η έξοδος θα είναι στη θέση x=0, y=0.

```
public class Maze {
	private int playerX;
	pricate int playerY;
	public boolean isAtExit() {
		return (playerX ==0 && playerY ==0);
	}
	public void setPosition(int x, int y) {
		playerX = x;
		playerY = y;
	}
}
```

Ο κώδικας δουλεύει τις περισσότερες φορές σωστά, αλλά ας φανταστούμε ότι ο παίκτης κινείται από το σημείο (1,0 στο (0,1). Γνωρίζουμε ότι οι μεταβλητές των αντικειμένων είναι playerX=1 και playerY=0, το νήμα A θα καλέσει την setPosition(0,1). Έπειτα το νήμα A θα εκτελέσει την πρώτη γραμμή κώδικα της μεθόδου θέτοντας τη playerX=0. Πριν όμως προχωρήσει στην επόμενη γραμμή, το νήμα B καλεί την isAtExit() και αφού εκείνη τη στιγμή ο παίκτης είναι στη θέση (0,0) επιστρέφει true. Για να λυθεί αυτό το πρόβλημα πρέπει να είμαστε σίγουροι ότι η μέθοδοι setPosition και isAtExit() δεν μπορούν να εκτελεστούν ταυτόχρονα.

Όπως αναφέραμε παραπάνω για να εμποδίζουμε δύο μεθόδους να τρέχουν ταυτόχρονα πρέπει να τις ορίσουμε ως synchronized:

```
public class Maze {
	private int playerX;
	pricate int playerY;
	public synchronized boolean isAtExit() {
		return (playerX ==0 && playerY ==0);
	}
	public synchronized void setPosition(int x, int y) {
		playerX = x;
		playerY = y;
	}
```

```
}
```

Ουσιαστικά τώρα όταν εκτελεί μία συγχρονισμένη μέθοδο, αποκτάει ταυτόχρονα μία κλειδαριά για αυτό το αντικείμενο. Μόνο μία κλειδαριά μπορεί να αποκτηθεί για ένα αντικείμενο κάθε φορά. Η κλειδαριά ξεκλειδώνει όταν η μέθοδος σταματάει να εκτελείται είτε επειδή έφτασε στο τέλος του κώδικα είτε επειδή πέταξε κάποια εξαίρεση. Οπότε αν μία συγχρονισμένη μέθοδο είναι κλειδωμένη, καμία άλλη συγχρονισμένη μέθοδος δεν μπορεί να τρέξει έως ότου απελευθερωθεί η κλειδωμένη μέθοδος.

Πέραν όμως τις συγχρονισμένες μεθόδους, μπορούμε επίσης να ορίσουμε συγχρονισμένα αντικείμενα επιτυγχάνοντας ουσιαστικά το ίδιο αποτέλεσμα. Με το συγχρονισμό αντικειμένου κλειδώνουμε μόνο το αντικείμενο μέσα στη μέθοδο, ενώ με το συγχρονισμό μεθόδου κλειδώνουμε ολόκληρη τη μέθοδο κάθε φορά. Οπότε στο παραπάνω παράδειγμα η μέθοδος setPosition θα γίνει:

```
public void setPosition(int x, int y) {
      synchronized(this) {
            playerX = x;
            playerY = y;
      }
}
```

Ο συγχρονισμός αντικειμένων είναι χρήσιμος όταν χρειαζόμαστε πάνω από μία κλειδαριά, όταν χρειαζόμαστε να αποκτήσουμε κλειδαριά εκτός από το this ή όταν δεν χρειάζεται να συγχρονίσουμε ολόκληρη τη μέθοδο.

Πότε όμως δεν πρέπει να συγχρονίζουμε τον κώδικα μας; Ο γενικός κανόνας λέει ότι δεν πρέπει να συγχρονίζουμε παραπάνω κώδικα από ότι πρέπει επειδή δημιουργεί αχρείαστες καθυστερήσεις στο πρόγραμμα μας όταν ένα ή περισσότερα νήματα προσπαθούν να εκτελέσουν το ίδιο συγχρονισμένο μπλοκ κώδικα. Για παράδειγμα, δεν χρειάζεται να συγχρονίσουμε όλη τη μέθοδο όταν μερικά σημεία της μεθόδους θέλουν συγχρονισμό. Αντί για αυτό τοποθετούμε ένα συγχρονισμένο μπλοκ γύρω από τα κρίσιμα σημεία του κώδικα μέσα στη μέθοδο. Επίσης δεν πρέπει να συγχρονίζουμε μεθόδους που χρησιμοποιούνε μόνο τοπικές μεταβλητές. Κάθε τοπική μεταβλητή αποθηκεύεται σε μία στοίβα και κάθε νήμα έχει τη δικιά της ξεχωριστεί στοίβα, οπότε δεν υπάρχει πιθανότητα να

δημιουργηθεί πρόβλημα συγχρονισμού. Τέλος δεν πρέπει να ανησυχούμε σχετικά με το αν πρέπει να συγχρονίσουμε τον κώδικα όταν δεν είναι προσπελάσιμος από πολλά νήματα. Παρά όλα αυτά θα πρέπει να ξέρουμε ότι κάποιος κώδικας χρησιμοποιείται μόνο από ένα νήμα ούτως ώστε να τον συγχρονίσουμε αν τυχόν υπάρξει αλλαγή του κώδικα στο μέλλον και πολλαπλά νήματα αποκτήσουν πρόσβαση σε αυτόν.

ΥΠΟΚΕΦΑΛΑΙΟ

2.3 ΑΔΙΕΞΟΔΟΣ (DEADLOCK)

Η αδιέξοδος (Deadlock) παρουσιάζεται όταν δύο νήματα σταματήσουν να λειτουργούν επειδή το ένα περιμένει το άλλο να κάνει κάτι. Ένα απλό παράδειγμα είναι:

1) Το νήμα Α αποκτά την κλειδαριά 1.
2) Το νήμα Β αποκτά την κλειδαριά 2.
3) Το νήμα Β περιμένει την κλειδαριά 1 να απελευθερωθεί.
4) Το νήμα Α περιμένει την κλειδαριά 2 να απελευθερωθεί.

Όπως παρατηρούμε τα δύο νήματα περιμένουν το ένα το άλλο και έτσι έχουμε αδιέξοδο. Αδιέξοδο προκαλείται μόλις πολλαπλά νήματα προσπαθούν να αποκτήσουν πολλαπλές κλειδαριές εκτός σε λάθος σειρά. Ο καλύτερος τρόπος για να αποφύγουμε το αδιέξοδο είναι να γράψουμε το συγχρονισμένο κώδικα με τέτοιο τρόπο ώστε να το αποφύγουμε. Πρέπει να σκεφτούμε ποια νήματα αποκτούν ποιες κλειδαριές και με τι σειρά. Υπάρχουν περιπτώσεις όπου το παιχνίδι μας σταματάει εξαιτίας ενός αδιέξοδου. Για να τα εντοπίσουμε γρήγορα και εύκολα μπορούμε να χρησιμοποιήσουμε κάποιο πρόγραμμα που να ανιχνεύει τα αδιέξοδα αυτόματα.

Η Java για να μπορέσει να πετύχει μία επικοινωνία μεταξύ των νημάτων και να αποτρέψει, όσο το δυνατόν αποτελεσματικά, τα τυχόν αδιέξοδα στις μεθόδους ή τα μέρη του κώδικα που έχουν δηλωθεί ως synchronized, δημιούργησε τις μεθόδους wait() και notify(). Η κλήση της μεθόδου wait() ενός αντικειμένου προκαλεί την αναστολή της εκτέλεσης του νήματος μέχρι ένα άλλο νήμα να καλέσει τη μέθοδο notify() ή notifyAll(). Για ένα αντικείμενο μπορεί να υπάρχει μια

ομάδα νημάτων τα οποία βρίσκονται σε αναστολή. Μόλις κληθεί η μέθοδος notify(), το πρώτο από τα νήματα που βρίσκονται σε αναστολή θα μεταπέσει σε κατάσταση ετοιμότητας κα από εκεί μπορεί να φθάσει σε κατάσταση εκτέλεσης. Όταν καλείται η μέθοδος notifyAll(), όλα τα νήματα τίθενται σε κατάσταση ετοιμότητας και άρα μπορούν να εκτελεστούν. Ο επόμενος κώδικας δείχνει τη χρήση της wait() και της notify():

```
public synchronized void waitForMessage() {   //Tread A
      try {
            wait();
      }
      catch (InterruptedException ex) { }
}
public synchronized void setMessage(String message) {   //Thread B
      …
      notify();
}
```

Ουσιαστικά εδώ το νήμα Α περιμένει το νήμα Β να εκτελεστεί και να καλέσει τη notify(). Μόλις το νήμα Β καλέσει τη notify(), απελευθερώνει την κλειδαριά και αφήνει το νήμα Α να συνεχίσει την εργασία του. Επίσης η wait() μπορεί να πάρει όρισμα το χρόνο αναμονής σε περίπτωση που ένα νήμα δεν καλέσει ποτέ τη notify(). Τέλος η μέθοδος notifyAll() ειδοποιεί όλα τα νήματα, αντί για ένα που κάνει η notify(), που περιμένουν την κλειδαριά να απελευθερωθεί.

ΥΠΟΚΕΦΑΛΑΙΟ
2.4 ΟΙ ΠΡΟΤΕΡΑΙΟΤΗΤΕΣ ΣΤΑ ΝΗΜΑΤΑ

Υπάρχουν περιπτώσεις που θέλουμε τα νήματα να μην είναι ισοδύναμα, αλλά κάποια από αυτά να εκτελούνται για περισσότερο χρόνο από τα άλλα. Τα νήματα που έχουν κάποια αλληλεπίδραση με το χρήστη πρέπει να έχουν μεγαλύτερη προτεραιότητα από άλλα νήματα που εκτελούνται στο παρασκήνιο. Η προτεραιότητα κάθε νήματος παίρνει τιμές από 1 μέχρι 10. Το μέγιστο 10 αντιστοιχεί στη μεγαλύτερη προτεραιότητα, το 1 αντιστοιχεί στην μικρότερη προτεραιότητα, ενώ το 5 είναι η κανονική προτεραιότητα. Για μεγαλύτερη ευκολία

η java.lang.Thread ορίζει τις 3 σταθερές: Thread.MAX_PRIORITY, Thread.MIN_PRIORITY, Thread.NORM_PRIORITY για τις προτεραιότητες 10, 1 και 5 αντίστοιχα. Αξίζει να πούμε ότι όσο μεγαλύτερη προτεραιότητα έχει ένα νήμα, τόσο περισσότερο χρόνο εκτέλεσης θα πάρει από την CPU. Τέλος οι δύο σημαντικότερες μέθοδοι που πρέπει να γνωρίζουμε είναι: η setPriority(int priority) η οποία θέτει μια προτεραιότητα σε ένα νήμα και η getPriority() η οποία επιστρέφει την προτεραιότητα ενός νήματος.

```
thread1.setPriority(Thread.MAX_PRIORITY);

thread2.setPriority(1);
```

ΕΠΙΛΟΓΟΣ

Στο κεφάλαιο αυτό αναφερθήκαμε στους τρόπους με του οποίου μπορούμε να συντάξουμε ένα νήμα στη Java, στην έννοια της λέξης κλειδί synchronized και πως μπορούμε να αποφύγουμε τα deadlocks. Στο επόμενο κεφάλαιο θα επικεντρωθούμε στη ανάπτυξη animation με Java και στην επίλυση του μεγαλύτερου προβλήματος των animation, του τρεμοπαιξίματος της οθόνης (frickering).

ΚΕΦΑΛΑΙΟ 3
ANIMATION

ΕΙΣΑΓΩΓΗ

Το animation (κινούμενες εικόνες) είναι η ταχεία προβολή μιας σειράς από εικόνες (δισδιάστατης ή τρισδιάστατης μορφής) ή θέσεων ενός μοντέλου, έτσι ώστε να δημιουργείται η ψευδαίσθηση της κίνησης. Ουσιαστικά δημιουργεί μια οπτική οφθαλμαπάτη της κίνησης και αυτό συμβαίνει εξ αιτίας του φαινομένου διατήρησης της εικόνας στο μάτι επί 1/12 του δευτερολέπτου (μεταίσθημα ή μετείκασμα). Κίνηση μπορεί να δημιουργηθεί και να παρουσιαστεί με πολλούς τρόπους. Η πιο διαδεδομένη μέθοδος απεικόνισης της κινούμενης εικόνας αποτελείται από ένα πρόγραμμα βίντεο ή κινουμένου σχεδίου.

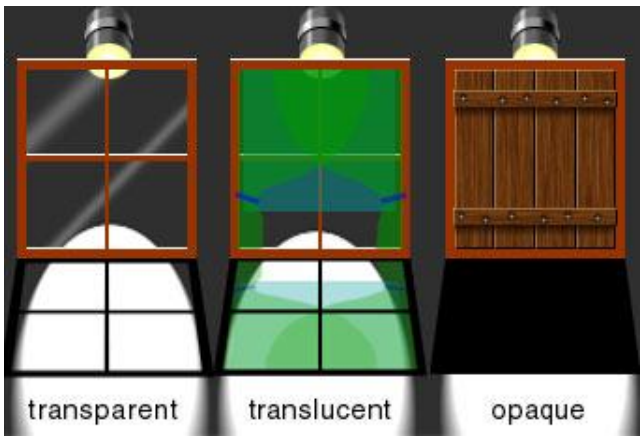
Οι εικόνες σε ένα animation ονομάζονται πλαίσια (frames). Κάθε πλαίσιο εμφανίζεται για ορισμένο χρονικό διάστημα, αλλά τα πλαίσια δεν είναι απαραίτητο να εμφανίζονται το ίδιο χρονικό διάστημα. Για παράδειγμα μπορούμε να ορίσουμε το πρώτο πλαίσιο να απεικονίζεται για 150 χιλιοστά του δευτερολέπτου, το δεύτερο πλαίσιο για 100, το τρίτο πλαίσιο για 200 και ούτω καθεξής.

ΥΠΟΚΕΦΑΛΑΙΟ
3.1 ΟΙ ΕΙΚΟΝΕΣ ΣΤΗ JAVA

Για να μπορέσουμε να δημιουργήσουμε ένα πρόγραμμα κινούμενων σχεδίων (Animation) πρέπει να γνωρίζουμε τα βασικά για τις εικόνες. Όταν θέλουμε να χρησιμοποιήσουμε μία εικόνα στο πρόγραμμά μας, ένα σημαντικό πράγμα που πρέπει να ξέρουμε είναι η διαφάνειά της (Σχήμα 12). Μπορούμε να χρησιμοποιήσουμε τρεις τύπου διαφάνειας:

- Αδιαφανής (Opaque): Κάθε pixel στην εικόνα μας είναι ορατό.
- Διαφανής (Transparent): Κάθε pixel στην εικόνα μας είναι είτε ορατό είτε διαφανές. Αυτό σημαίνει ότι το άσπρο φόντο μπορεί να είναι διαφανές και να μην το βλέπουμε όταν σχεδιάζεται η εικόνα.

- Ημιδιαφανής (Translucent): Τα pixels μπορεί να είναι μερικώς διαφανή δημιουργώντας στην εικόνα ένα ημιδιαφανές αποτέλεσμα.



Σχήμα 12 "Οι τρεις τύποι διαφάνειας"

Για παράδειγμα, στην εικόνα του χαρακτήρα που θέλουμε να εισάγουμε στο παιχνίδι ή animation μας, πιθανότατα θέλουμε να μετατρέψουμε το άσπρο φόντο της εικόνας διαφανές έτσι ώστε να εμφανίζεται μόνο το σώμα του χαρακτήρα και όχι το άσπρο φόντο γύρω του. Στις περισσότερες εικόνες της πτυχιακή εργασίας έγινε χρήση του προγράμματος CorelDRAW 12 με σκοπό το φόντο των εικόνων να γίνει διαφανές.

Οι δύο βασικοί τύποι εικόνας είναι η εικόνα pixel (raster) και η εικόνα διανύσματος (vector). Η εικόνα pixel αποτελείται από ένα πλέγμα pixels στο οποίο το κάθε pixel μπορεί να έχει το διαφορετικό χρώμα ή σκίαση. Η εικόνα διανύσματος περιγράφει μία εικόνα γεωμετρικά και μπορεί να αλλάξει μέγεθος χωρίς να υποβαθμίζει την ποιότητα. Η Java API έχει επικεντρωθεί στις εικόνες pixel από ότι στις εικόνες διανύσματος. Οι τρεις βασικοί τύποι εικόνων pixel που υποστηρίζει η Java και δεν χρειάζεται κάποια ιδιαίτερη προσπάθεια για να διαβαστούν είναι: ο τύπος GIF(Graphics Interchange Format), ο τύπος JPEG(Joint Photographics Expert Groups) και ο τύπος PNG(Portable Network Graphics).

Πως μπορούμε όμως να μεταφράσουμε ένα αρχείο GIF, PNG ή JPEG έτσι ώστε να μπορεί να εμφανιστεί στην οθόνη μας; Αυτό γίνεται με τη χρήση της μεθόδου getImage() την οποία παρέχει η Toolkit. Αυτή η μέθοδος αναλύει το αρχείο της εικόνας και επιστρέφει ένα αντικείμενο εικόνας. Για παράδειγμα:

```
Toolkit toolkit = Toolkit.getDefaultToolkit();
```

```
Image image = toolkit.getImage(filename);
```

Αυτός ο κώδικας φαίνεται αθώος αρκετά, αλλά στην πραγματικότητα δεν φορτώνει την εικόνα. Η εικόνα ξεκινάει να φορτώνεται σε ένα άλλο νήμα, οπότε αν προσπαθήσουμε να εμφανίσουμε την εικόνα πριν προλάβει να τη φορτώσει ολόκληρη, τότε μόνο ένα μέρος της εικόνας θα απεικονιστεί. Μπορούμε να χρησιμοποιήσουμε το αντικείμενο MediaTracker να παρακολουθήσουμε την εικόνα και να περιμένουμε να φορτωθεί, αλλά υπάρχει ευκολότερη λύση. Η χρήση της κλάσης ImageIcon φορτώνει την εικόνα χρησιμοποιώντας τον MediaTracker για εμάς. Η κλάση ImageIcon ανήκει στο πακέτο της javax.swing και οι ενέργειες που κάνει είναι να φορτώνει την εικόνα χρησιμοποιώντας το Toolkit και να περιμένει να φορτωθεί η εικόνα προτού την επιστρέψει. Για παράδειγμα:

```
ImageIcon icon = new ImageIcon(filename);

Image image = icon.getImage();
```

Για να εμφανίζουμε μια εικόνα σε ένα applet, πρέπει πρώτα να τη φορτώσουμε από το δίσκο ή από την περιοχή του δικτύου στην οποία βρίσκεται. Για να γίνει αυτό, πρέπει να δημιουργήσουμε ένα αντικείμενο της κλάσης Image η οποία ανήκει στο πακέτο java.awt και επιπλέον να δημιουργήσουμε ένα αντικείμενο της κλάσης UML το οποίο ανήκει στο πακέτο java.net. Το URL αντικείμενο είναι ουσιαστικά ένα αλφαριθμητικό το οποίο δηλώνει τη θέση του αρχείου της εικόνας στο διαδίκτυο ή στον τοπικό δίσκο του υπολογιστή μας. Για παράδειγμα η δημιουργία αντικειμένων URL γίνεται ως εξής:

```
URL location = new URL("http://τοποθεσία_εικόνας");

URL location = new URL("C:\Pictures\ateithe_logo.jpeg");
```

Μια καλύτερη μέθοδος για να δημιουργήσουμε το αντικείμενο URL με τη θέση της εικόνας είναι η χρήση των μεθόδων getCodeBase() και getDocumentBase(). Η μέθοδος getDocumentBase() επιστρέφει τη θέση URL όπου βρίσκεται το τρέχον αρχείο .html που αναφέρεται στο applet.

```
URL location = getDocumentBase();
```

Οπότε για να φορτώσουμε την εικόνα χρησιμοποιούμε το αντικείμενο URL μαζί με τη σχετική θέση της εικόνας. Σε περίπτωση που θέλουμε να μεταφέρουμε αρχεία

μας σε άλλο φάκελο, τότε η συνάρτηση getDocumentBase() θα επιστρέφει τη νέα θέση χωρίς να αλλάξουμε τίποτα στον κώδικα του applet.

Η μέθοδος getCodeBase() επιστρέφει ένα αλφαριθμητικό με τη θέση URL, στην οποία βρίσκεται ο φάκελος που περιέχει το applet.

```
URL location = getCodeBase();
```

Αυτή η μέθοδος είναι χρήσιμη όταν αποθηκεύουμε τις εικόνες μας στον ίδιο φάκελο ή σε κάποιο υποφάκελο του αρχείου .class του applet. Έτσι όταν καλούμε τη μέθοδο getCodeBase() παίρνουμε τη θέση της εικόνας. Αν τυχόν μεταφέρουμε τα αρχεία από ένα φάκελο σε ένα άλλο, δεν θα χρειαστεί να αλλάξουμε καθόλου τον κώδικα του applet.

Για να μπορέσουμε να φορτώσουμε μία εικόνα σε ένα applet κάνοντας χρήση των URL, πρέπει αρχικά να δημιουργήσει ένα αντικείμενο URL με τη τοποθεσία της εικόνας και έπειτα να καλέσει την getImage() για να φορτώσει την εικόνα. Η μέθοδος getImage() μπορεί να γραφεί με δύο μορφές:

- Image image = getImage(URL url_εικόνας);

  π.χ.
  ```
  URL location = new URL("http://τοποθεσία_εικόνας");
  ```
  ή
  ```
  URL location = new URL("C:\Pictures\ateithe_logo.jpeg");

  Image image = getImage(location);
  ```
- Image image = getImage(URL url_καταλόγου, String μονοπάτι_εικόνας);
  Εδώ μπορούμε να εφαρμόσουμε τις μεθόδους getDocumentBase() και getCodeBase() που αναφερθήκαμε παραπάνω:

  ```
  URL base = getDocumentBase();   ή   URL base = getCodeBase();

  Image image = getImage(base, "Images/ateithe_logo.jpeg");
  ```

ΥΠΟΚΕΦΑΛΑΙΟ
3.2 ANIMATION ΜΕ JAVA

Για να μπορέσουμε να φτιάξουμε ένα animation σε Java πρέπει να κάνουμε χρήση της μεθόδου paint() σε ένα applet. Η κλήση της μεθόδου paint() γίνεται όταν

σχεδιάζεται αρχικά το applet ή όταν πρέπει να ξανασχεδιαστεί κατά τη μετακίνηση του παραθύρου του στην περίπτωση που μετακινηθεί ένα κάποιο άλλο παράθυρο που το κάλυπτε. Επίσης μπορεί και ο προγραμματιστής να την καλέσει όταν θέλει να εισάγει ένα νέο περιεχόμενο στο applet. Με λίγα λόγια η δουλειά της μεθόδου paint() είναι να τοποθετεί εικόνες στο applet μας. Εάν λοιπόν έχουμε μια ομάδα από κατάλληλες εικόνες, μπορούμε να τις προβάλλουμε τη μία μετά την άλλη, ρυθμίζοντας τη διάρκεια κάθε εικόνας ώστε να δώσουμε την αίσθηση της κίνησης.

Πως όμως μπορούμε να συντάξουμε ένα πρόγραμμα το οποίο θα προβάλει τα πλαίσια (frames) το ένα μετά το άλλο στην οθόνη για ορισμένο χρονικό διάστημα; Η απάντηση είναι απλή. Για να μπορέσουμε να ενημερώσουμε την οθόνη πολλές φορές το δευτερόλεπτο, χρειάζεται να δημιουργήσουμε ένα νήμα το οποίο θα περιέχει τον animation βρόχο. Ο animation βρόχος είναι υπεύθυνος να κρατάει τη σωστή «πορεία» των πλαισίων και να ενημερώνει περιοδικά την οθόνη. Η εφαρμογή ενός νήματος στο πρόγραμμά μας, όπως είπαμε στο προηγούμενο κεφάλαιο, γίνεται είτε γράφοντας μία κλάση η οποία να απευθύνεται στην κλάση Thread είτε εφαρμόζοντας την διασύνδεση Runnable. Ένα συχνό λάθος που γίνεται είναι η τοποθέτηση του animation βρόχου μέσα στη μέθοδο paint(). Αυτό το λάθος δημιουργεί παράξενα αποτελέσματα στο πρόγραμμά μας επειδή το κεντρικό νήμα AWT αναλαμβάνει δύο εργασίες, οι οποίες είναι να σχεδιάζει πάνω στο applet και να διαχειρίζεται τα γεγονότα. Έτσι λοιπόν αφού ο animation βρόχος θα βρίσκεται εκτός της μεθόδου paint(), θα πρέπει να καλούμε μια μέθοδο η οποία θα δίνει το έναυσμα ώστε η Java να καλέσει τη μέθοδο paint() και να ξανασχεδιάσει το applet. Η μέθοδος αυτή είναι η repaint().

Έτσι λοιπόν όπως είπα παραπάνω, για να υπάρξει κίνηση στις εικόνες μας πρέπει να δημιουργήσουμε ένα νήμα. Το νήμα είναι απαραίτητο σε εργασίες που εκτελούνται συνεχώς και χρειάζονται πολύ από το χρόνο του επεξεργαστή. Οι κινούμενες εικόνες είναι ένα παράδειγμα μια τέτοια εργασίας. Το πρώτο πράγμα που πρέπει να κάνουμε για να συντάξουμε ένα animation είναι να ορίσουμε τις μεθόδους start(), run() και stop(). Η μέθοδος start() δημιουργεί ένα νήμα στο applet μας και η μέθοδος stop() αναστέλλει τη εκτέλεση του applet. Η μέθοδος run() παίζει το σημαντικότερο ρόλο στο animation μας επειδή ουσιαστικά είναι η μέθοδος που τοποθετούμε το animation βρόχο μας, έτσι ώστε να δημιουργήσουμε κίνηση. Αυτός ο βρόχος ακολουθεί τα παρακάτω βήματα:

1) Ενημερώνει το animation.
2) Σχεδιάζει πάνω στην οθόνη.
3) Προαιρετικά καλεί την μέθοδο sleep() για σύντομο χρονικό διάστημα.
4) Επιστρέφει στο βήμα 1.

Το πρώτο βήμα ενημερώνει το animation με την «πορεία» των πλαισίων που ακολουθούν. Έπειτα το δεύτερο βήμα καλεί τη repaint() η οποία σχεδιάζει τα νέα πλαίσια πάνω στην οθόνη. Το βήμα 3 είναι προαιρετικό και έχει σχέση με το αν θέλουμε να προσθέσουμε μια πολύ σύντομη καθυστέρηση ανάμεσα στην προβολή των πλαισίων. Τέλος το βήμα 4 είναι όταν ο βρόχος φτάνει στο τέλος του και επιστρέφει στην αρχή για να ξαναεκτελέσει τα βήματα 1 μέχρι 3.

Στο επόμενο παράδειγμα δημιουργείται ένα animation που απεικονίζει τον Duke, τη μασκότ της Java, να μας χαιρετάει (Σχήμα3):

```
import java.awt.*;
import java.applet.*;

public class AnimationExample extends Applet implements Runnable {

    Thread thread;
    Image duke_waves[] = new Image[10];
    Image current_duke, background;
    //Ορίζει ένα πίνακα με το χρόνο που θα διακόπτει το νήμα πριν
εμφανίσει την επόμενη εικόνα
    int delay[] = {200, 200, 175, 175, 150, 150, 175, 175, 200, 200};

    public void init() {
     //Αρχικοποιεί τον πίνακα εικόνων, το φόντο και το μέγεθος του applet
        for (int i=0; i<duke_waves.length;i++){
            duke_waves[i]            =            getImage(getCodeBase(),
"Examples/Images/duke" + (i+1) + ".gif");
        }

        background               =               getImage(getCodeBase(),
"Examples/Images/java_background.jpg");

        this.setSize(500, 300);
    }
```

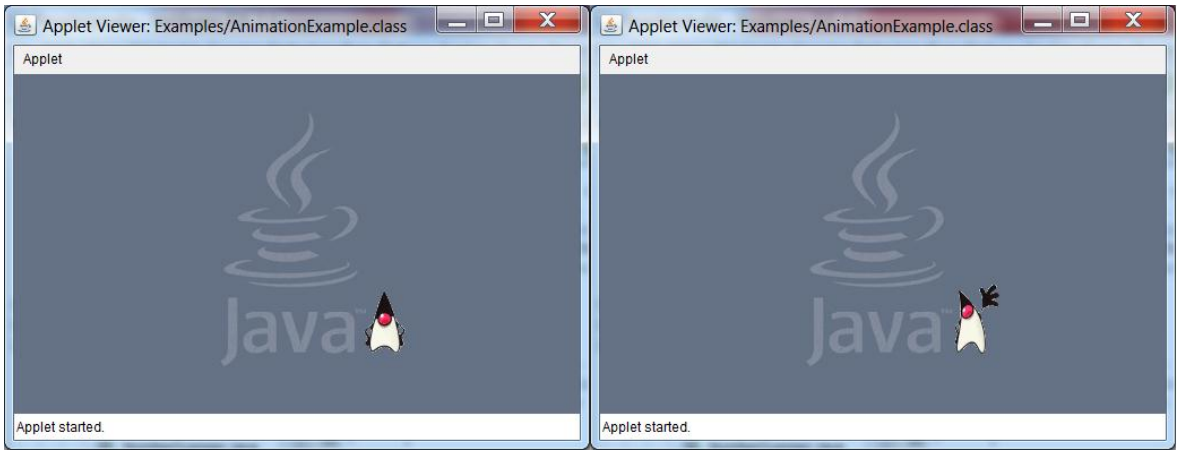
```
    public void start() {
        //Δημιουργία και εκτέλεση του νήματος
        if (thread==null){
            thread=new Thread(this);
            thread.start();
        }
    }

    public void stop(){
        //Αναστέλλει την εκτέλεση του applet
        if (thread!=null){
            thread=null;
        }
    }

    public void run() {
      try{
            int frame = 0;

            while (true){
                //Animation βρόχος
                current_duke = duke_waves[frame%10];
                repaint();
               //Διακόπτει το νήμα για λίγα χιλιοστά του δευτερολέπτου
                thread.sleep(delay[frame%10]);
                frame++;
            }
        }
        catch (InterruptedException e) {}
    }

    public void paint(Graphics g) {
        //Σχεδίαση του φόντου
        g.drawImage(background, 0, 0, 500, 300, this);
        //Σχεδίαση της εικόνας
        g.drawImage(current_duke, 300, 180, this);
    }
}
```

Στη μέθοδο paint() κάνουμε χρήση της μεθόδου drawImage η οποία σχεδιάζει μία εικόνα πάνω στο applet μας. Όμως οι δύο μέθοδοι διαφέρουν ως προς τις παραμέτρους. Στη σχεδιάση του φόντου χρησιμοποιούμε την:

`drawImage(Image εικόνα, int x, int y, int μήκος, int ύψος, ImageObserver παρατηρητής)` – Ορίζει το σημείο χ και ψ που θα εμφανιστεί η εικόνας στο applet, καθώς και το μήκος και το πλάτος της εικόνας. Ο ImageObserver παίρνει τιμή null.

Ενώ στη σχεδίαση της εικόνας έχουμε τη μέθοδο:

`drawImage(Image εικόνα, int x, int y, ImageObserver παρατηρητής)` - Ορίζει το σημείο χ και ψ που θα εμφανιστεί η εικόνα στο applet. Ο ImageObserver παίρνει τιμή null.



Σχήμα 13 "Η απεικόνιση του 1ου και 6ου πλαισίου"

ΥΠΟΚΕΦΑΛΑΙΟ
3.3 ΤΟ ΠΡΟΒΛΗΜΑ ΤΟΥ ΤΡΕΜΟΠΑΙΞΙΜΑΤΟΣ

Ένα σοβαρό πρόβλημα που μπορούμε να παρατηρήσουμε στο πρόγραμμα AnimationExample είναι ότι ο χαρακτήρας μας duke τρεμοσβήνει (fricker) κατά την κίνηση του. Αυτό έχει ως αποτέλεσμα να κάνει κακή αίσθηση και να ενοχλεί αυτούς που παρακολουθούν το animation.

Γιατί συμβαίνει αυτό και πως μπορούμε να ξεφορτωθούμε αυτό το πρόβλημα; Αυτό συμβαίνει επειδή συνεχώς σχεδιάζουμε κατευθείαν πάνω στην

οθόνη. Πιο συγκεκριμένα, σβήνουμε το χαρακτήρα με το φόντο και μετά ξανασχεδιάζουμε το χαρακτήρα, οπότε υπάρχουν φορές που βλέπουμε για σύντομο χρονικό διάστημα το φόντο στο σημείο όπου ο χαρακτήρας έπρεπε να βρίσκεται. Επειδή αυτό γίνεται πολύ γρήγορα, εμφανίζεται ως τρεμοπαίξιμο στην οθόνη. Η λύση στο πρόβλημα αυτό είναι να χρησιμοποιήσουμε την διπλή προσωρινή αποθήκευση (double buffering).

Ένας buffer είναι μία απλή περιοχή μνήμης προσωρινής αποθήκευσης εκτός της οθόνης η οποία χρησιμοποιείται για σχεδίαση. Όταν έχουμε διπλή προσωρινή αποθήκευση αντί να σχεδιάζουμε απευθείας πάνω στην οθόνη, σχεδιάζουμε πρώτα σε ένα buffer και έπειτα αντιγράφουμε ολόκληρη την εικόνα του buffer στην οθόνη μας χωρίς να ζωγραφίζουμε λίγο-λίγο επάνω. Με αυτόν τον τρόπο ολόκληρη η οθόνη μας ενημερώνεται αμέσως και μπορούμε να δούμε καθαρά την κίνηση του χαρακτήρα χωρίς να τρεμοσβήνει. Ο buffer μας μπορεί να είναι μία κανονική εικόνα.

Αν θέλουμε να χρησιμοποιήσουμε διπλό buffer στο applet αυτό που πρέπει να κάνουμε είναι δημιουργήσουμε μία μέθοδο update() η οποία θα ορίζει ένα πλαίσιο εκτός οθόνης με το μέγεθος του πλαισίου της οθόνης, θα σβήνει κάθε φορά την εικόνα του πλαισίου εκτός οθόνης, θα καλεί την paintFrame() για να σχεδιάσει την εικόνα πάνω στο πλαίσιο εκτός οθόνης και τέλος θα αντιγράφει ολόκληρη την εικόνα που έχουμε πλαίσιο εκτός οθόνης στο πλαίσιο της οθόνης μας. Για να κατασκευάσουμε ένα buffer μπορούμε να κάνουμε χρήση της μεθόδου createImage(int μήκος, int ύψος). Οπότε αυτό που πρέπει να κάνουμε στο πρόγραμμα AnimationExample για να αποφύγουμε το πρόβλημα του τρεμοπαιξίματος είναι να ορίσουμε τρεις νέες μεταβλητές, να προσθέσουμε δύο νέες μεθόδους, την update() και την paintFrame(), και να τροποποιήσουμε την paint().

```
Dimension offDimension;
Image offImage;
Graphics offGraphics;
…
public void update(Graphics g) {
      Dimension d = size();

      // Δημιουργεί το offscreen γραφικό περιεχόμενο.
```

```
        if ((offGraphics == null) || (d.width != offDimension.width) ||
(d.height != offDimension.height)) {
            offDimension = d;
            offImage = createImage(d.width, d.height);
            offGraphics = offImage.getGraphics();
        }

        // Σβήνει την προηγούμενη εικόνα του offscreen πλαισίου
        offGraphics.setColor(getBackground());
        offGraphics.fillRect(0, 0, d.width, d.height);
        offGraphics.setColor(Color.black);

        // Σχεδιάζει τις εικόνες μέσα στο offscreen πλαίσιο
        paintFrame(offGraphics);
        //Σχεδιάζει την εικόνα πάνω στην οθόνη
        g.drawImage(offImage, 0, 0, null);
}

public void paint(Graphics g) {
        update(g);
}

public void paintFrame(Graphics g) {
          //Σχεδίαση του φόντου
          g.drawImage(background, 0, 0, 500, 300, this);
          //Σχεδίαση της εικόνας
          g.drawImage(current_duke, 300, 180, this);
}
```

ΥΠΟΚΕΦΑΛΑΙΟ
3.4 Η ΧΡΗΣΗ ΤΟΥ MEDIATRACKER

Ένα πρόγραμμα της Java μπορεί όταν φορτώνει μία εικόνα, να την εμφανίζει στην οθόνη πριν προλάβει να τη φορτώσει ολόκληρη. Η κλάση MediaTracker χρησιμοποιείται για να προσδιορίσει πότε μια εικόνα έχει φορτωθεί πλήρως. Η MediaTracker γνωρίζει ποιες εικόνες είναι ήδη φορτωμένες στη μνήμη και έχει τη δυνατότητα να εμποδίζει τη χρήση μιας εικόνας η οποία δεν έχει φορτωθεί εξ ολοκλήρου. Ένα άλλο πλεονέκτημα είναι ότι το πρόγραμμα μπορεί να εκτελεί άλλες εργασίες ταυτόχρονα ενώ φορτώνει την εικόνα.

Στα animation προγράμματα η MediaTracker είναι χρήσιμη στις εικόνες φόντου, οι οποίες έχουν μεγάλο μέγεθος και θέλουν περισσότερο χρόνο να φορτωθούν. Ωστόσο μπορεί να αποσπάσει το χρόνο εκτέλεσης του προγράμματος χωρίς κανένα όφελος όταν χρησιμοποιείται για μικρές εικόνες. Επομένως είναι στην κρίση του προγραμματιστή αν θέλει να περιμένει να φορτώσει όλο το animation προτού το εμφανίσει στην οθόνη.

Για να κατασκευάσουμε ένα αντικείμενο MediaTracker γράφουμε:

```
MediaTracker tracker = new MediaTracker(this);
```

Το όρισμα του κατασκευαστή είναι ένα ανικείμενο ImageObserver. Στο αντικείμενο ImageObserver εμφανίζεται πάνω η εικόνα, έτσι βάζουμε τη λέξη-κλειδί this επειδή θέλουμε να εμφανίσουμε την εικόνα πάνω στο applet.

Η προσθήκη των εικόνων στον Tracker μας γίνεται με τη χρήση της μεθόδου: `addImage(Image image, int id)` η οποία παίρνει ορίσματά την εικόνα που θέλουμε να εισάγουμε και ένα id αριθμό που αναφέρεται σε μια εικόνα ή σε μια συλλογή εικόνων. Έτσι λοιπόν στο παράδειγμα AnimatioExample θα προσθέσουμε στον if βρόχο όπου ορίζουμε τις εικόνες μας τον κώδικα:

```
for (int i=0; i<duke_waves.length;i++){
      duke_waves[i] = getImage(getCodeBase(), "Examples/Images/duke" +
(i+1) + ".gif");
      tracker.addImage(duke_waves[i], 0);
}

background                    =                    getImage(getCodeBase(),
"Examples/Images/java_background.jpg"
tracker.addImage(duke_waves[i], 0);
```

Έπειτα για να ελέγξουμε αν όλες οι εικόνες έχουν φορτωθεί πλήρως, έτσι ώστε να προχωρήσουμε στη σχεδίαση τους πάνω στην οθόνη, κάνουμε χρήση της μεθόδου: `statusID(int id, boolean load`). Στο όρισμα id βάζουμε τον αριθμό αναγνώρισης της εικόνας και στο load αν το ορίζουμε true τότε ξεκινάει να φορτώνει τις εικόνες που δεν έχει φορτώσει ακόμη. Η μέθοδος statusID επιστρέφει ουσιαστικά την κατάσταση της εικόνας με το συγκεκριμένο id. Οι καταστάσεις που μπορεί να επιστρέψει είναι: LOADING (όταν ακόμα φορτώνει την εικόνα),

ABORTED (όταν έχει σταματήσει την φόρτωση για κάποιο λόγο), ERRORED (όταν έχει υπάρξει κάποιο σφάλμα), και COMPLETE (όταν έχει ολοκληρώσει με επιτυχία την φόρτωση της εικόνας). Επομένως η σχέση που θα βάλουμε στη μέθοδο paintFrame του παράδειγμος AnimatioExample για να ελέγχει αν ο MediaTracker φόρτωσε όλες τις εικόνες είναι:

```
public void paintFrame(Graphics g) {
      if (tracker.statusID(0, true) == MediaTracker.COMPLETE) {
            g.drawImage(background, 0, 0, 500, 300, this);
            g.drawImage(current_duke, 300, 180, this);
      }
}
```

Σε περίπτωση που θέλουμε το πρόγραμμα να περιμένει μέχρι να φορτωθεί πλήρως μία εικόνα, κατά τη διάρκεια καταγραφής των εικόνων στον tracker, τότε εφαρμόζουμε τη μέθοδο: `waitForID(int id)` με όρισμα το id της εικόνας που αναμένουμε. Επίσης υπάρχουν και οι μέθοδοι: `waitForAll()` στην οποία το πρόγραμμα περιμένει μέχρις ότου όλες οι εικόνες φορτωθούν στον tracker, και `waitForID(int id, long ms)` η οποία έχει την ίδια λειτουργία με τη μέθοδο `waitForID(int id)` με τη διαφορά ότι παίρνει δεύτερο όρισμα το χρόνο αναμονής σε msec του προγράμματος. Στο παράδειγμα AnimatioExample μπορούμε να προσθέσουμε στη μέθοδο init() την παρακάτω γραμμή κώδικα:

```
try{
      tracker.waitForID(0);
}
catch(InterruptedException ie) {}
```

ΕΠΙΛΟΓΟΣ

Στο κεφάλαιο αυτό έγινε εκτενής αναφορά στη δημιουργία αντικειμένων εικόνων και animation προγραμμάτων. Επίσης διατυπώθηκε η σημαντικότητα των νημάτων στα animations και πως μπορούμε να αποφύγουμε το φαινόμενο του τρεμοπαιξίματος της οθόνης κατά την προβολή του animation. Στο επόμενο κεφάλαιο θα αναλύσουμε τα γεγονότα στη Java και θα δούμε πόσο χρήσιμα είναι στα παιχνίδια.

ΚΕΦΑΛΑΙΟ 4
ΤΑ ΓΕΓΟΝΟΤΑ

ΕΙΣΑΓΩΓΗ

Ας φανταστούμε ένα παιχνίδι όπου ο παίκτης δεν χρειάζεται να κάνει καμιά κίνηση, όπως να πατήσει ένα κουμπί, και όλες οι κινήσεις να γίνονται χωρίς τις ενέργειες του παίκτη. Στην πραγματικότητα αυτό το παιχνίδι δεν θα υπήρχε ή απλά θα ήταν πολύ βαρετό διότι θα ήταν ένα παιχνίδι χωρίς διαδραστικότητα (interactivity). Η διαδραστικότητα ουσιαστικά υπάρχει όταν σε ένα πρόγραμμα ο χρήστης εισάγει δεδομένα και αμέσως το πρόγραμμα αλλάζει τα στοιχεία πάνω στην οθόνη σχετικά με την εισαγωγή δεδομένων. Ο χρήστης μπορεί να δώσει εντολές μέσω του ποντικιού ή του πληκτρολογίου.

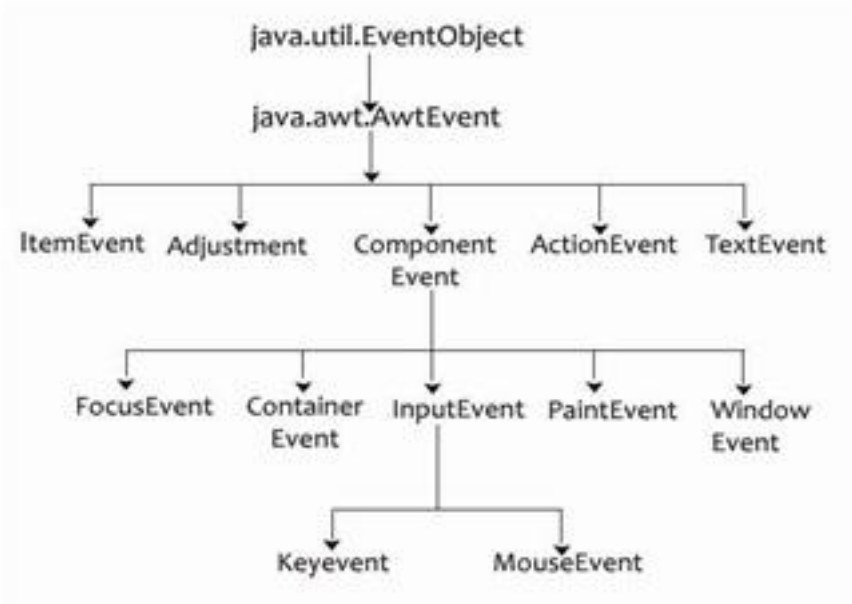
Η δημιουργία ενός παιχνιδιού σε Java απαιτεί την ύπαρξη προγραμματισμένων κουμπιών όπου κάθε φορά που ο χρήστης πατάει ένα κουμπί να συμβαίνει το ανάλογο γεγονός. Το γεγονός έχει ως αποτέλεσμα για παράδειγμα την κίνηση του χαρακτήρα στην οθόνη. Όπως καταλαβαίνουμε τα καινούργια προγράμματα, σε σύγκριση με τα προγράμματα παλιότερης τεχνολογίας, εξαρτούνται από τις επιλογές του χρήστη για να μπορέσουν να εκτελεστούν. Για να μπορέσει η Java να επεξεργαστεί τέτοιου είδους καταστάσεις χρησιμοποιεί από το JDK 1.1 και μετά το μοντέλο αποστολή γεγονότων (delegation event model) σε συνεργασία με το σύνολο γραφικών βιβλιοθηκών JFC (Java Foundation Classes). Οι βιβλιοθήκες JFC εμπεριέχουν τις βιβλιοθήκες AWT και Swing, το Java 2D API, καθώς και κάποιες επιπλέον τεχνολογίες όπως Pluggable Look and Feel, Drag and Drop, κτλ.

Όταν συμβαίνει ένα γεγονός, όπως αναφερθήκαμε στην προηγούμενη παράγραφο ο χρήστης αλληλεπιδρά με κάποιο από τα στοιχεία μιας γραφικής εφαρμογής που υπάρχει σε ένα applet κινώντας το ποντίκι, πατώντας ένα κουμπί ή επιλέγοντας ένα στοιχείο μιας λίστας, δημιουργείται ένα αντικείμενο γεγονότος (event object). Το αντικείμενο γεγονότος περιέχει πληροφορίες για το ίδιο το γεγονός και για την πηγή από την οποία προέρχεται. Επίσης το αντικείμενο αυτό ψάχνει να βρει μια ειδική μέθοδο η οποία θα χειριστεί το γεγονός αυτό. Οι κλάσεις

των γεγονότων για τη βιβλιοθήκη AWT βρίσκονται στο πακέτο java.awt.event. Ο προγραμματιστής μπορεί να τοποθετήσει στο πρόγραμμα μεθόδους χειρισμού γεγονότων έτσι ώστε το applet και τα διάφορα συστατικά που περιέχει (μενού, κουμπιά, πεδία κειμένου κ.λπ.) να αποκρίνονται αλληλεπιδραστικά με το χρήστη. Υπάρχουν διάφορες κατηγορίες γεγονότων, μερικές από τις οποίες είναι:

- ActionEvent: Δημιουργείται από κάποια ενέργεια του χρήστη, όπως πάτημα σε ένα κουμπί, διπλοπάτημα σε κάποιο μενού, κ.τ.λ.
- MouseEvent: Δημιουργείται όταν ο χρήστης μετακινεί το ποντίκι ή πατάει κάπου με αυτό.
- MouseMotionAdapter: Δημιουργείται ένα γεγονός όταν ο δείκτης του ποντικιού κινείται ή σέρνεται.
- KeyEvent: Δημιουργείται όταν πιέζεται ή απελευθερώνεται ένα πλήκτρο του πληκτρολογίου.
- ComponentEvent: Δημιουργείται ένα συστατικό το οποίο αποκρύπτεται, εμφανίζεται, μετακινείται, ή αλλάζει μέγεθος

Υπάρχουν περιπτώσεις όπου μπορεί να δημιουργηθούν περισσότεροι από ένα τύποι γεγονότων. Για παράδειγμα αν πατήσουμε ένα κουμπί μπορεί να δημιουργηθεί ένα ActionEvent, καθώς επίσης και ComponentEvent, FocusEvent, MouseEvent και KeyEvent. Η ιεραρχία κλάσεων γεγονότων φαίνεται στο Σχήμα 14:



Σχήμα 14 " Η ιεραρχία κλάσεων γεγονότων "

Για να μπορέσει ένα γεγονός να γίνει αντιληπτό, πρέπει να ορίζουμε έναν ακροατή γεγονότος (event listener). Ο ακροατής γεγονότος είναι ένα αντικείμενο μιας κλάσης, η οποία υλοποιεί μια ή περισσότερες διασυνδέσεις του πακέτου java.awt.event. Το αντικείμενο αυτό έχει τη δυνατότητα να «ακούει» το γεγονός που παράγεται μόλις πατήσουμε ένα κουμπί στο πληκτρολόγιο, ποντίκι, κ.λπ. Τότε καλείται ο χειριστής γεγονότων, μια μέθοδος που αντιστοιχεί σε κάποιο τύπο γεγονότος. Κάθε διασύνδεση ακροατή παρέχει μία ή περισσότερες μεθόδους χειρισμού.

ΥΠΟΚΕΦΑΛΑΙΟ
4.1 ΓΕΓΟΝΟΤΑ ΠΛΗΚΤΡΟΛΟΓΙΟΥ

Όταν θέλουμε να προγραμματίσουμε ένα παιχνίδι στο οποίο είναι απαραίτητη η χρήση πολλών πλήκτρων, όπως τα βελάκια για την κίνηση του χαρακτήρα και το πλήκτρο SpaceBar για να πυροβολεί ο χαρακτήρας, τότε χρησιμοποιούμε το KeyEvent.

Για να αιχμαλωτίσουμε ένα γεγονός πλήκτρου, πρέπει να κάνουμε δύο πράγματα: να δημιουργήσουμε ένα KeyListener και να καταχωρήσουμε τον ακροατή γεγονότος για να πάρουμε ένα γεγονός. Η καταχώρηση του ακροατή γεγονότος γίνεται απλά με το κάλεσμα της μεθόδου addKeyListener() στο συστατικό που θέλουμε να δεχθεί γεγονός πλήκτρου. Για παράδειγμα:

```
Component.addKeyListener(this);
```

Αυτό που πρέπει να κάνουμε έπειτα για να κατασκευάσουμε ένα KeyListener είναι να δημιουργήσουμε ένα αντικείμενο το οποίο εφαρμόζει τη διασύνδεση KeyListener. Η διασύνδεση KeyListener έχει τρεις μεθόδους:

- keyPressed() – το γεγονός συμβαίνει όταν ο χρήστης πατάει το πλήκτρο.
- keyReleased() - το γεγονός συμβαίνει όταν ο χρήστης απελευθερώνει το πλήκτρο.
- keyTyped() - το γεγονός συμβαίνει όταν ο χρήστης πατάει το κουμπί και μετά βασίζεται στον επαναληπτικό ρυθμό πατήματος αυτού του κουμπιού. Στα παιχνίδια το keyTyped() δεν είναι χρήσιμο όσο το keyPressed() και keyTyped().

Και οι τρεις μέθοδοι που αναφέραμε παίρνουν ως παράμετρο ένα αντικείμενο KeyEvent. Το αντικείμενο KeyEvent μας βοηθάει να ανιχνεύσουμε το κουμπί που πάτησε ή απελευθέρωσε ο χρήστης, παρέχοντας μας μια ψηφιακή μορφή κώδικα για κάθε πλήκτρο (virtual key code). Ο ψηφιακός κώδικας πλήκτρου είναι ένα ιδιαίτερο πληκτρολόγιο που ορίστηκε από τη Java και διαφέρει ως προς τους χαρακτήρες του από το κανονικό. Για παράδειγμα, παρόλα που το A και το a είναι διαφορετικοί χαρακτήρες, έχουν τον ίδιο κώδικα πλήκτρου. Πιο συγκεκριμένα η γενική μορφή του ψηφιακού κώδικα πλήκτρου είναι VK_xxx. Με αυτό τον τρόπο μπορούμε να μαντέψουμε τον κώδικα των περισσότερων πλήκτρων, όπως το πλήκτρο G αντιστοιχεί στον κώδικα KeyEvent.VK_G και το πλήκτρο ENTER στον κώδικα KeyEvent.VK_ ENTER. Στον παρακάτω κώδικα ο χρήστης μπορεί μέσω των κουμπιών W, S, A και D να κινήσει το διαστημόπλοιο πάνω, κάτω, δεξιά και αριστερά αντίστοιχα στο πλαίσιο.

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class KeyPressedExample extends JFrame implements KeyListener {
    Container cont;
    JLabel ss = new JLabel(new
ImageIcon("src/Examples/Images/spaceship.PNG"));

    public KeyPressedExample(){
        super("KeyPressed Example");
        this.setVisible(true);
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        this.setSize(500,500);
        cont = getContentPane();
        cont.add(ss);
        cont.setLayout(null);
        ss.setBounds(250, 250, 64, 64);
        addKeyListener(this);  //Προσθέτουμε ένα KeyListener στο πλαίσιο
    }
    public void keyTyped(KeyEvent e) {}
    public void keyPressed(KeyEvent e) {
    //Με τα πλήκτρα W, S, A και D κινούμε το spaceship στο JFrame
        if (e.getKeyCode() == KeyEvent.VK_W)
            ss.setBounds(ss.getX(), ss.getY()-10, 64, 64);
```

```
        if (e.getKeyCode() == KeyEvent.VK_S)
            ss.setBounds(ss.getX(), ss.getY()+10, 64, 64);
        if (e.getKeyCode() == KeyEvent.VK_A)
            ss.setBounds(ss.getX()-10, ss.getY(), 64, 64);
        if (e.getKeyCode() == KeyEvent.VK_D)
            ss.setBounds(ss.getX()+10, ss.getY(), 64, 64);
    }
    public void keyReleased(KeyEvent e) {}

    public static void main(String Args[]){
        new KeyPressedExample();
    }
}
```

ΥΠΟΚΕΦΑΛΑΙΟ

4.2 ΓΕΓΟΝΟΤΑ ΠΟΝΤΙΚΙΟΥ

Το πληκτρολόγιο είναι απλά ένα θεμελιώδης πίνακας με ένα πλήθος κουμπιών πάνω του, αλλά το ποντίκι είναι μια πολυπλοκότερη συσκευή. Όχι απλά το ποντίκι έχει πλήκτρα (εξαρτάται από το μοντέλο του ποντικιού, μπορεί να έχει δύο, τρία, ή περισσότερα πλήκτρα), αλλά διαθέτει επίσης κίνηση και πιθανότατα έχει ροδέλα κύλισης περιεχομένων. Με τη χρήση του ποντικιού μπορούμε να πάρουμε τρία διαφορετικά γεγονότα:

- Από τα κλικ που γίνονται με τα κουμπιά του ποντικιού.
- Από την κίνηση του ποντικιού.
- Από την ροδέλας κύλισης περιεχομένων.

Τα πλήκτρα του ποντικιού συμπεριφέρονται σαν τα πλήκτρα του πληκτρολογίου, με τη διαφορά ότι δεν χρησιμοποιούν την επανάληψη πλήκτρου. Η κίνηση του ποντικιού ορίζεται από την x και y συντεταγμένη. Τέλος, τα γεγονότα της ροδέλας κύλισης περιεχομένων δείχνουν πόσο μακριά κύλισε τη ροδέλα ο χρήστης.

Κάθε γεγονός ποντικιού έχει το δικό του ακροατή: MouseListener για τα πλήκτρα του ποντικιού, MouseMotionListener για την κίνηση του ποντικιού και MouseWheelListener για τη ροδέλα του ποντικιού.

ΥΠΟΚΕΦΑΛΑΙΟ
4.3 Η ΔΙΑΣΥΝΔΕΣΗ KEYLISTENER

Όπως τη διασύνδεση KeyListener του πληκτρολογίου έτσι και η διασύνδεση MouseListener έχει μεθόδους για την ανίχνευση της πίεσης, της απελευθέρωσης και του κλικ του πλήκτρου του ποντικιού. Η μέθοδοι του MouseListener είναι:

- mousePressed(MouseEvent e) – Ενεργοποιείται όταν ένα κουμπί του ποντικιού πιέζεται πάνω σε ένα συστατικό.
- mouseReleased(MouseEvent e) - Ενεργοποιείται όταν ένα κουμπί του ποντικιού απελευθερώνεται πάνω σε ένα συστατικό.
- mouseClicked(MouseEvent e) – Ενεργοποιείται όταν ένα κουμπί του ποντικιού πιέζεται και απελευθερώνεται πάνω σε ένα συστατικό.
- mouseEntered(MouseEvent e) - Ενεργοποιείται όταν ένα κουμπί του ποντικιού μπαίνει σε ένα συστατικό.
- mouseExited(MouseEvent e) - Ενεργοποιείται όταν ένα κουμπί του ποντικιού βγαίνει από ένα συστατικό.

Όπως η keyTyped(), έτσι και η mouseClicked() δεν είναι χρήσιμη όταν θέλουμε να φτιάξουμε ένα παιχνίδι. Για να βρούμε ποιο πλήκτρο πιέστηκε ή απελευθερώθηκε κάθε φορά καλούμε την μέθοδο getButton() της MouseEvent.

ΥΠΟΚΕΦΑΛΑΙΟ
4.4 Η ΔΙΑΣΥΝΔΕΣΗ MOUSEMOTIONLISTENER

Η διασύνδεση MouseMotionListener μπορεί να ανιχνεύσει δύο τύπους κίνησης: την συνηθισμένη κίνηση και την κίνηση συρσίματος ποντικιού. Οι μέθοδοι που έχει η MouseMotionListener για τον εντοπισμό αυτών των γεγονότων είναι:

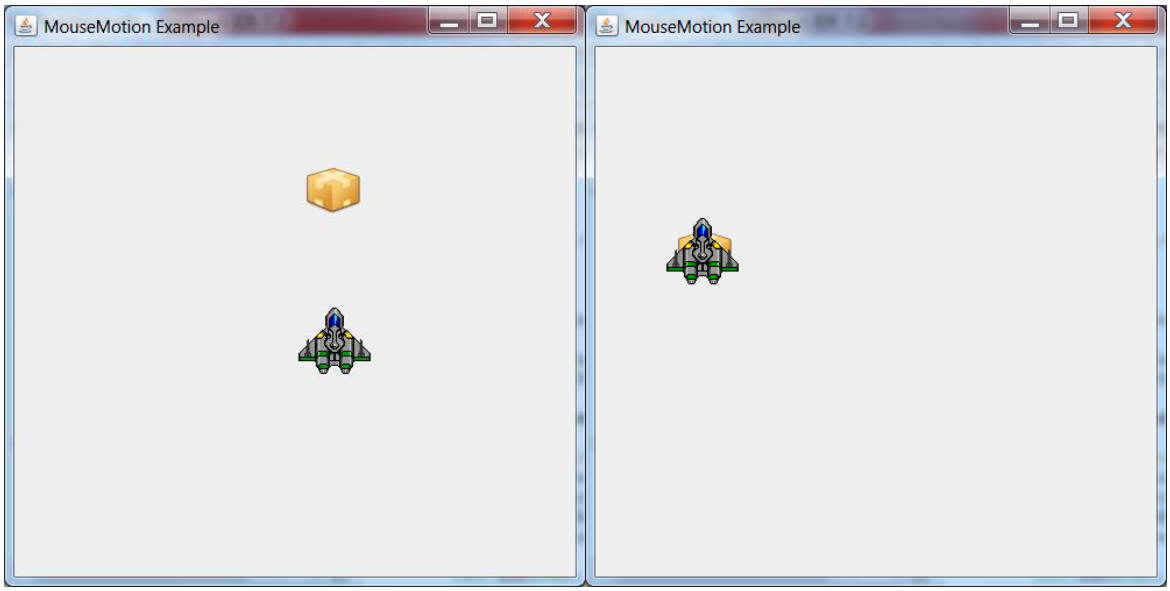
- mouseDragged(MouseEvent e) – Επικαλείται όταν κάποιο κουμπί του ποντικιού πιέζεται πάνω σε ένα συστατικό και έπειτα σέρνει αυτό κάπου αλλού.
- mouseMoved(MouseEvent e) – Επικαλείται όταν ο δείκτης του ποντικιού κινείται προς ένα συστατικό χωρίς να έχει πιεστεί κάποιο κουμπί.

Ο εντοπισμός των τρέχων συντεταγμένων x και y του ποντικιού, οποιαδήποτε τύπου κίνησης, γίνονται με το κάλεσμα των μεθόδων getX() και getY() της MouseEvent.

Στο επόμενο παράδειγμα ο χρήστης μπορεί να κινήσει ένα διαστημόπλοιο μέσα στο πλαίσιο με τη βοήθεια της μεθόδου mouseMoved(). Ουσιαστικά αυτό που γίνεται είναι να δηλώνουμε κάθε φορά που κινείται ο δείκτης του ποντικιού, τις νέες συντεταγμένες στην ετικέτα μας. Ακόμα έχει προστεθεί ένα κουτί στο πλαίσιο όπου ο χρήστης με τη γνωστή ενέργεια «σύρε και άφησε» (drag and drop) μπορεί με πατημένο ένα κουμπί του ποντικιού να πάρει το κουτί και να το μεταφέρει σε κάποιο άλλο σημείο του πλαισίου απελευθερώνοντας το κουμπί (Σχήμα 15). Όλη αυτή η διαδικασία γίνεται χάρης τη μέθοδο mouseDragged().

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class MouseMotionExample extends JFrame implements
MouseMotionListener {
    Container cont;
    JLabel ss = new JLabel(new
ImageIcon("src/Examples/Images/spaceship.PNG"));
    JLabel box = new JLabel(new
ImageIcon("src/Examples/Images/box.PNG"));

    public MouseMotionExample()
    {
        super("MouseMotion Example");
        this.setVisible(true);
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        this.setSize(500,500);
        cont = getContentPane();
        cont.add(ss);
        cont.add(box);
        cont.setLayout(null);
        ss.setBounds(250, 250, 64, 64);
        box.setBounds(250, 100, 48, 48);
        addMouseMotionListener(this);
    }
```

```
    public void mouseDragged(MouseEvent e) {
        //Ορίζει νέες συντεταγμένες στο ss και box κάθε φορά που σύρεται,
δημιουργώντας την αίσθηση ότι το διαστημόπλοιο μεταφέρει το κουτί.
        box.setBounds(e.getX()-35, e.getY()-50, 48, 48);
        ss.setBounds(e.getX()-45, e.getY()-60, 64, 64);
    }
    public void mouseMoved(MouseEvent e) {
        //Ορίζει τις συντεταγμένες του δείκτη του ποντικιού στο
διαστημόπλοιο κάθε φορά που κινούμε το ποντίκι.
        ss.setBounds(e.getX()-35, e.getY()-50, 64, 64);
    }

    public static void main(String Args[]){
        new MouseMotionExample();
    }
}
```



Σχήμα 15 "Το αποτέλεσμα της εκτέλεσης του MouseMotionExample"

ΥΠΟΚΕΦΑΛΑΙΟ
4.5 Η ΔΙΑΣΥΝΔΕΣΗ MOUSEWHEELLISTENER

Η διασύνδεση MouseWheelListener χρησιμοποιεί μία υποκλάση της MouseEvent η οποία ονομάζεται MouseWheelevent. Η μόνη μέθοδος που έχει η MouseWheelListener είναι η mouseWheelMoved(MouseWheelEvent e) η οποία ενεργοποιείται όταν η ροδέλα του ποντικιού περιστρέφεται. Για να εξετάσουμε πόσο πολύ περιστράφηκε η ροδέλα του ποντικιού καλούμε τη μέθοδο

getWheelRotation(). Αν η μέθοδος επιστρέψει αρνητική τιμή τότε σημαίνει ότι κινηθήκαμε προς τα πάνω και αν η μέθοδος επιστρέψει θετική τιμή τότε σημαίνει ότι κινηθήκαμε προς τα κάτω.

ΥΠΟΚΕΦΑΛΑΙΟ
4.6 ΤΑ ΣΗΜΑΣΙΟΛΟΓΙΚΑ ΓΕΓΟΝΟΤΑ

Οι κλάσεις των ακροατών παρακολουθούν δύο τύπου γεγονότων: τα γεγονότα χαμηλού επιπέδου και τα σημασιολογικά γεγονότα. Στην πρώτη κατηγορία ανήκουν τα συμβάντα ποντικιού και πληκτρολογίου που αναφέραμε παραπάνω. Στη δεύτερη κατηγορία ανήκουν «ανώτερα» γεγονότα όπως το πάτημα σε ένα κουμπί διαταγής η οποία έχει αποτέλεσμα να εκτελεστεί ένα κομμάτι κώδικα, να επιλέγει ένα στοιχείο μενού, να χρησιμοποιηθεί μία ράβδος κύλισης, κ.λπ. Οι ακροατές που παρακολουθούν σημασιολογικά γεγονότα είναι ο ActionListener, ο AdjustmentListener και ο ItemListener.

Η διασύνδεση ActionListener έχει μία μέθοδο, τη μέθοδο ActionPerformed(ActionEvent e). Η μέθοδος αυτή επικαλείται όταν συμβαίνει μία ενέργεια στο πρόγραμμα μας. Την ενέργεια αυτή μπορεί να την κάνει ο χρήστης όταν πατάει σε μία περιοχή κειμένου, επιλέγει μία ετικέτα ή ένα μενού, πατάει κάποιο κουμπί πλαισίου, κ.λπ. Για να προσθέσουμε στο συστατικό μας ένα τέτοιο γεγονός καλούμε τη μέθοδο addActionListener().

Η δεύτερη διασύνδεση AdjustmentListener περιλαμβάνει και αυτή μία μέθοδο, την adjustmentValueChanged για το χειρισμό των γεγονότων. Για να καλεστεί αυτή η μέθοδος πρέπει να έχει αλλάξει μία ρυθμιζόμενη τιμή στο πρόγραμμα μας. Για παράδειγμα μπορούμε να την εφαρμόσουμε στις μπάρες κύλισης των πλαισίων. Η μέθοδος σύνδεσης με την πηγή αυτού του γεγονότος είναι η addAdjustmentListener().

Τέλος, η διασύνδεση ItemListener έχει την itemStateChanged(ItemEvent e) για τη διαχείριση των γεγονότων της. Η μέθοδος καλείται όταν ο χρήστης επιλέγει ή αποεπιλέγει ένα αντικείμενο στο πρόγραμμα, για παράδειγμα ένα κουτάκι επιλογής (checkbox). Για να συνδέσουμε ένα συστατικό με αυτό το γεγονός κάνουμε χρήση της μεθόδου addItemListener().

ΕΠΙΛΟΓΟΣ

Σε αυτό το κεφάλαιο αναλύσαμε τα γεγονότα που παράγονται όταν πατάμε ένα κουμπί στο πληκτρολόγιο ή στο ποντίκι, καθώς και όταν κινούμε το ποντίκι. Επίσης αναφέραμε το ρόλο τους στα παιχνίδια και γενικότερα στα προγράμματα που χρειάζονται την ενέργεια του χρήστη για να προχωρήσουν. Στο επόμενο κεφάλαιο θα δώσουμε πληροφορίες για το πως μπορούμε να δημιουργήσουμε αρχεία ήχου και πώς να εμπλουτίζουμε το παιχνίδι μας με αυτά.

ΚΕΦΑΛΑΙΟ 5
ΗΧΗΤΙΚΑ ΕΦΕ ΚΑΙ ΜΟΥΣΙΚΗ

ΕΙΣΑΓΩΓΗ

Είναι λογικό ένα παιχνίδι να διαθέτει πλούσια ηχητικά εφέ και μερικές φορές και μουσική για να εντυπωσιάσει και να διασκεδάσει τον παίκτη. Όταν παίζουμε ένα παιχνίδι, μπορούμε να ακούμε τα ηχητικά εφέ αλλά στην πραγματικότητα να μην τα προσέχουμε. Αυτό συμβαίνει επειδή περιμένουμε να τα ακούσουμε, ωστόσο κάποιες φορές τα ηχητικά εφέ που παράγονται όταν χάνουμε γίνονται πιο αισθητά σε εμάς. Για να συνοψίσουμε, τα ηχητικά εφέ είναι σημαντικό μέρος του των παιχνιδιών, επειδή οι παίκτες περιμένουν να τα ακούσουν και δεν πρέπει να τα παραλείπουμε από τα παιχνίδια.

ΥΠΟΚΕΦΑΛΑΙΟ
5.1 ΤΟ JAVA SOUND API

Για να παίξουμε ένα δείγμα ήχου στο Java παιχνίδι μας, απλά χρησιμοποιούμε το Java Sound API που βρίσκεται στο πακέτο javax.sound.sampled. Το Java Sound API μπορεί να παίξει μορφές ήχου όπως 8 ή 16 bit δείγματα, καθώς και δείγματα ήχου με συχνότητα από 8000Hz μέχρι 48.000Hz. Επίσης, μπορεί να παίξει μονό ή στέρεο ήχο. Ο ήχος του παιχνιδιού μας εξαρτάται από τις προτιμήσεις μας, για παράδειγμα μπορούμε να χρησιμοποιήσουμε 16-bit, μονό, 44.000Hz ήχο.

Στις παλιότερες εκδόσεις της Java, τα αρχεία ήχου που υποστήριζε η Java ήταν μόνο με προέκταση .au. Τώρα όμως υπάρχει μία τυποποιημένη επέκταση στο Java API που λέγεται JMF (Java Media Framework). Η JMF είναι μία βιβλιοθήκη της Java, η οποία επιτρέπει αρχεία ήχου, εικόνας και βίντεο να προστεθούν σε μία Java εφαρμογή ή applet. Αυτό το πακέτο, το οποίο μπορεί να αιχμαλωτίσει, να παίξει, να μεταγλωττίσει κώδικα και να δημιουργήσει ένα ρεύμα πολλαπλών μορφών ήχου και άλλων μέσων. Με αυτό τον τρόπο η Java SE (Standard Edition) επεκτείνεται και επιτρέπει την ανάπτυξη νέων εφαρμογών πολυμέσων. Η JMF αναγνωρίζει τρεις τύπους ήχου που κυκλοφορούν: AIFF, AU

και WAV. Και οι τρεις αυτοί τύποι είναι ευέλικτοι και δεν έχει μεγάλη σημασία ποιο από αυτούς χρησιμοποιούμε.

ΥΠΟΚΕΦΑΛΑΙΟ
5.2 ΔΗΜΙΟΥΡΓΙΑ ΑΡΧΕΙΟΥ ΗΧΟΥ

Όπως στις εικόνες, έτσι και στα αρχεία χρησιμοποιούμε τις μεθόδους getDocumentBase() ή getCodeBase της κλάσης Applet για να προσδιορίσουμε τη URL διεύθυνση του αρχείου που περιέχει τον κώδικα .html ή .java αντίστοιχα. Οπότε αν έχουμε αποθηκεύσει το αρχείο ήχου στο φάκελο που βρίσκεται το .html ή java αρχείο, τότε απλά χρησιμοποιούμε τις μεθόδους αυτές. Με αυτόν τον τρόπο δεν χρειάζεται να τροποποιήσουμε τον κώδικα όταν αλλάζουμε την τοποθεσία του πηγαίου αρχείου στο δίσκο.

```
URL location = getDocumentBase();     //Για html αρχείο.
URL location = getCodeBase();         //Για java αρχείο.
```

Αν δεν θέλουμε να ορίσουμε απευθείας τη διαδρομή του αρχείου γράφουμε:

```
URL location = new URL("τοποθεσία_ήχου");
```

Στο επόμενο βήμα πρέπει να δημιουργήσουμε ένα αρχείο Audioclip στο οποίο θα ορίσουμε την τοποθεσία URL του αρχείου ήχου. Για να το κάνουμε αυτό μπορούμε να χρησιμοποιήσουμε μία από τις δύο μεθόδους getAudioClip της Applet:

```
AudioClip sound = getAudioClip(URL url);
AudioClip sound = getAudioClip(URL url, String όνομα_αρχείου);
```

Έτσι λοιπόν για να μπορέσουμε να αναπαράγουμε ή να σταματήσουμε ένα ήχο καλούμε τις μεθόδους του αντικειμένου Audioclip:

- loop() – Επαναλαμβάνει το παίξιμο του αρχείου ήχου.
- play() – Παίζει το αρχείο ήχου μία φορά.
- stop() – Σταματάει το παίξιμο του ήχου.

Επιπλέον η Applet παρέχει δύο γρήγορες μεθόδους για την αναπαραγωγή ενός ήχου χωρίς να χρειάζεται να δημιουργήσουμε URL και AudioClip αντικείμενα:

```
play(URL url);
```

```
play(URL url, String όνομα_αρχείου);
```

Με τη χρήση των παραπάνω μεθόδων ο προγραμματιστής μπορεί εύκολα να εισάγει στο παιχνίδι όλους τους ήχους που είναι απαραίτητοι για να γίνει πιο ρεαλιστικό και ικανοποιητικό προς τον παίκτη.

ΥΠΟΚΕΦΑΛΑΙΟ
5.3 3D ΗΧΟΣ

Ο 3D ήχος, ονομαζόμενος και ως κατευθυνόμενη ακοή, δημιουργεί μία πλουσιότερη εμπειρία στους παίκτες με την τοποθέτηση ψηφιακής πηγής ήχου στο 3D χώρο. Ήχους σαν και αυτόν κάνουν το παιχνίδι πιο ρεαλιστικό και προσφέρουν μία επιπλέον διάσταση. Για παράδειγμα ο παίκτης μπορεί να ακούει έναν εχθρό ο οποίος τον πλησιάζει αθόρυβα από πίσω του, καθώς και τον ήχο μιας πόρτας που ανοίγει από μακριά.

Πολλά διαφορετικά εφέ χρησιμοποιούνται για να παραχθεί ένας 3D ήχος. Μερικά από αυτά είναι:

- Η ένταση του ήχου ελαττώνεται κάθε φορά που ο παίκτης απομακρύνεται από την πηγή του.
- Ο ήχος που βγαίνει από το δεξί ηχείο είναι δυνατότερος από το αριστερό ηχείο όταν η πηγή βρίσκεται δεξιά από τον παίκτη και το αντίθετο.
- Εφαρμογή ηχητικών εφέ δωματίου όπου τα κύματα ήχου αναπηδάνε στους τοίχους και παράγουν ηχώ και αντήχηση.
- Εφαρμογή του φαινομένου Doppler στα ηχητικά εφέ του παιχνιδιού. Για παράδειγμα, ο ήχος ενός αντικειμένου που κινείται προς τον παίκτη έχει μεγαλύτερη ένταση από ένα σταματημένο αντικείμενο.

ΕΠΙΛΟΓΟΣ

Σε αυτό το κεφάλαιο αναφέραμε τον τρόπο με τον οποίο μπορούμε να προσθέσουμε ηχητικά εφέ και μουσική στο παιχνίδι ή animation μας. Επίσης εξηγήσαμε όλες τις μεθόδους της AudioClip που χειρίζονται το αρχείο ήχου. Στο επόμενο και τελευταίο κεφάλαιο θα αναλύσουμε τη Java 3D και όλων των απαραίτητων συστατικών της για τη δημιουργία τρισδιάστατων γραφικών.

ΚΕΦΑΛΑΙΟ 6
JAVA 3D

ΕΙΣΑΓΩΓΗ

Η Java 3D API (Application Programming Interface) είναι μια διασύνδεση προγραμματισμού εφαρμογών η οποία χρησιμοποιείται για την σύνταξη 3D εφαρμογών και applets. Αυτό προσφέρει στους προγραμματιστές ένα πλήθος υψηλού επιπέδου δομών για την κατασκευή και το χειρισμό 3D γεωμετρικών σχημάτων. Οι προγραμματιστές εφαρμογών μπορούν να απεικονίσουν ψηφιακούς κόσμους χρησιμοποιώντας αυτές τις δομές, τις οποίες παρέχει η Java 3D με αρκετές πληροφορίες για να αποδώσουν τους ψηφιακούς κόσμους όσο το δυνατόν πιο αποτελεσματικά.

Ένα όφελος της Java 3D είναι ότι τα προγράμματα 3D γραφικών μπορούν να τρέξουν σε οποιαδήποτε υπολογιστή και λειτουργικό σύστημα. Αυτό συμβαίνει διότι η Java 3D είναι μέρος της JavaMedia, επιτρέποντας της να τρέχει σε ένα ευρύ σύνολο συστημάτων. Επίσης αυτή συγχωνεύεται με το διαδίκτυο επειδή οι εφαρμογές και τα applets που χρησιμοποιούνε Java 3D API έχουν πρόσβαση σε ολόκληρο το σύνολο των Java κλάσεων.

Η Java 3D API αντλεί τις ιδέες της από τα υπάρχοντα API και από νέες τεχνολογίες. Οι χαμηλού επιπέδου δομές της Java 3D συνθέτουν τις καλύτερες ιδέες που βρέθηκαν σε χαμηλού επιπέδου API όπως Direct3D, OpenGL, QuickDraw3D, και XGL. Παρομοίως, οι υψηλού επιπέδου δομές της συνθέτουν τη καλύτερη ιδέα που βρέθηκε σε ορισμένα συστήματα βασισμένα με γραφικά. Ακόμα η Java 3D παρουσιάζει μερικές έννοιες οι οποίες συνήθως δεν θεωρούνται μέρος του γραφικού περιβάλλοντος, όπως ο 3D ήχος χώρου. Οι δυνατότητες του ήχου της Java 3D βοηθάνε στην παροχή εμπειριών του χρήστη.

ΥΠΟΚΕΦΑΛΑΙΟ
6.1 ΟΙ ΣΤΟΧΟΙ ΤΗΣ JAVA 3D

Η Java 3D σχεδιάστηκε να έχει πολλούς στόχους στο νου της. Ο βασικότερος στόχος της είναι η υψηλή απόδοση. Έτσι χρειάστηκε να παρθούν

πολλές αποφάσεις κατά τη διάρκεια της σχεδίασης της Java 3D έτσι ώστε να προσφέρουν τη μέγιστη απόδοση στους χρήστες των εφαρμογών.

Άλλοι στόχοι της Java 3D είναι:

- Παρέχει ένα πλούσιο σύνολο χαρακτηριστικών για τη δημιουργία ενδιαφέρον 3D κόσμων.
- Παρέχει ένα υψηλού επιπέδου αντικειμενοστραφούς προγραμματισμού πρότυπο το οποίο δίνει τη δυνατότητα στους προγραμματιστές να αναπτύξουν πολύπλοκες εφαρμογές και applets γρήγορα.
- Παρέχουν υποστήριξη για φορτωτές εκτέλεσης. Αυτό επιτρέπει τη Java 3D να προσαρμόσει μια ευρεία ποικιλία από μορφές αρχείων, όπως CAD μορφές, εναλλασσόμενες μορφές, VRML 1.0, και VRML 2.0.

ΥΠΟΚΕΦΑΛΑΙΟ

6.2 DIRECTX ΄Η OPENGL

Η απάντηση στο ερώτημα για το ποια έκδοση της Java 3D είναι καλύτερη απαιτεί τη διερεύνηση των σχετικών πλεονεκτημάτων του OpenGL έναντι του DirectX Graphics.

Όσον αναφορά τις τεχνικές λεπτομέρειες, τα δύο APIs είναι σχεδόν ισοδύναμα καθώς βασίζονται στην ίδια ιδέα, εκείνης της αρχιτεκτονικής της διασωλήνωσης γραφικών. Η πιο σημαντική τους διαφορά εντοπίζεται στην φορητότητά τους (portability), με την OpenGL από τη μία μεριά να καλύπτει ένα μεγάλο εύρος από πλατφόρμες και λειτουργικά συστήματα και το directX να περιορίζεται αποκλειστικά σε υπολογιστές Windows , καθώς και στην παιχνιδομηχανή Xbox. Το DirectX ελέγχεται αποκλειστικά από τη Microsoft, ενώ το OpenGL Architecture Review Board (ARB) αποτελεί μια ανοικτή συνεργασία πολλών εφαρμογών. Η DirectX Graphics έκδοση της Java 3D είναι διαθέσιμη μόνο για Windows.

ΥΠΟΚΕΦΑΛΑΙΟ

6.3 ΔΟΜΗ JAVA 3D ΠΡΟΓΡΑΜΜΑΤΟΣ

Αυτό το κεφάλαιο δείχνει πως ένας προγραμματιστής μπορεί να συντάξει μία Java 3D εφαρμογή και απεικονίζει την ιεραρχία μιας απλής Java 3D

εφαρμογής (Σχήμα 1). Πιο συγκεκριμένα θα αναλύσουμε με ποια σειρά πρέπει να δημιουργήσουμε τα αντικείμενα και να τα εφαρμόσουμε σε ένα Scene Graph, καθώς και ποιος κόμβος είναι γονιός (parent) και ποιος παιδί (child).

Ο γράφος σκηνής αποτελείται από διάφορα μέρη τα οποία είναι: ένα VirtualUniverse αντικείμενο και ένα Locale αντικείμενο, τα οποία βρίσκονται στην κορυφή του δέντρου, και ένα σύνολο από κλαδιά γράφων.

Ο κόμβος VirtualUniverse είναι ο κορυφαίος κόμβος σε ένα γράφο σκηνής και αντιπροσωπεύει τον εικονικό κόσμο και το σύστημα αναφοράς αυτού. Κάτω από το VirtualUniverse βρίσκεται ο κόμβος Locale ο οποίος καθορίζει τις συντεταγμένες τριών διαστάσεων που θα έχουν τα κλαδιά γράφοι (Branch Groups). Επίσης ένα VirtualUniverse μπορεί να περιέχει όσους Locale κόμβους χρειαστεί. Για παράδειγμα ένα αντικείμενο Locale μπορεί να οριστεί στις συντεταγμένες (0.0, 0.0, 0.0).

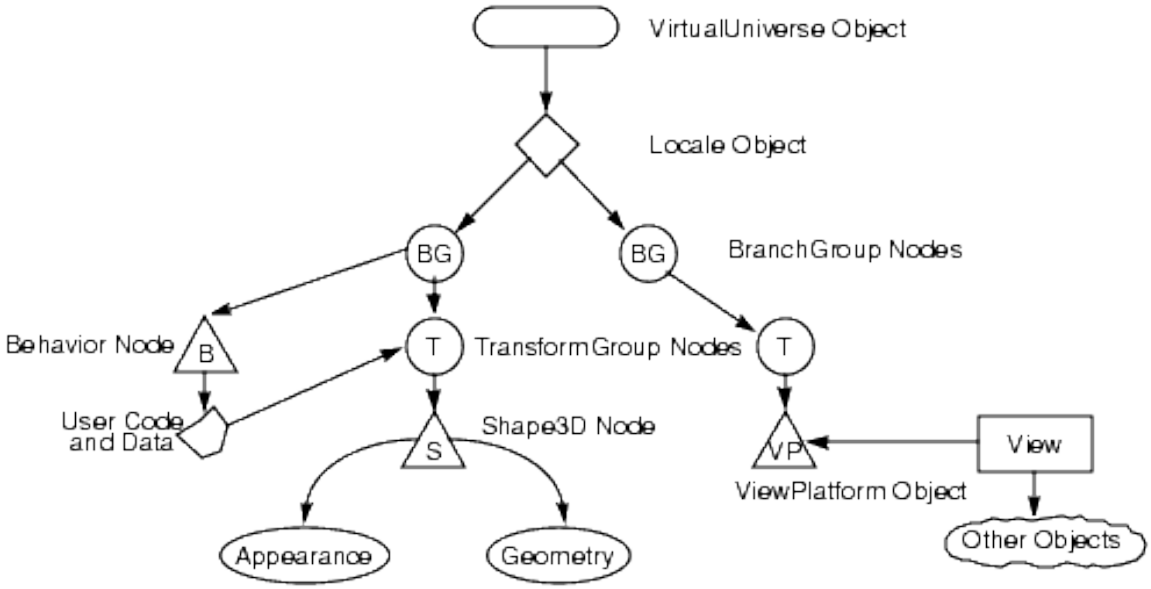
Όπως αναφέραμε παραπάνω το Locale έχει 2 κλαδιά τα οποία ονομάζονται Branch Groups.

Το αριστερό BranchGroup περιέχει 2 κλαδιά. Το πρώτο κλαδί είναι ο κόμβος συμπεριφοράς (Behavior Leaf Node), ο οποίος σχετίζεται με τη γεωμετρία του αντικειμένου. Το δεύτερο κλαδί του αριστερού BranchGroup αποτελείται από ένα TranformationGroup κόμβο ο οποίος σχετίζεται με τη θέση, τις συντεταγμένες και το μέγεθος του αντικειμένου στο Virtual Universe. Κάτω ακριβώς από αυτόν βρίσκεται ο Shape3D κόμβος ο οποίος αποτελείται από το Appearance αντικείμενο και το Geometry αντικείμενο. Το Geometry αναφέρεται στο τι γεωμετρικό σχήμα είναι το αντικείμενο μας, ενώ το Appearance περιγράφει το χρώμα, την υφή, τον φωτισμό κ.λπ.

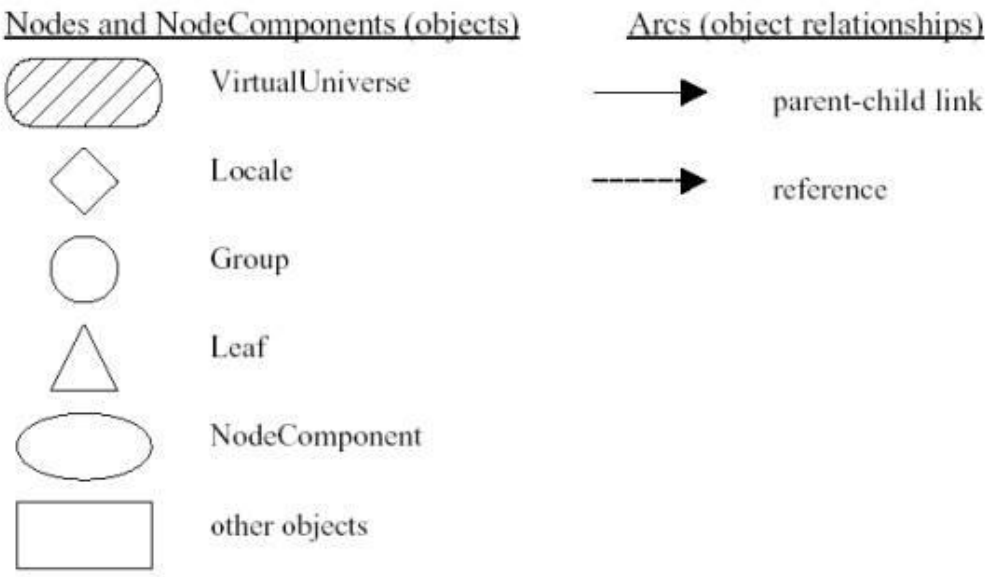
Το δεξί BranchGroup αποτελείται από ένα TransformGroup κόμβο και ένα ViewPlatform κόμβο φύλλου. Το TransformGroup καθορίζει τη θέση, τις συντεταγμένες, και το μέγεθος του ViewPlarform. Το αντικείμενο ViewPlatform ουσιαστικά ορίζει το οπτικό πεδίο που θα έχει στο τέλος ο χρήστης μέσα στο Virtual Universe. Ακόμα το ViewPlatform αναφέρεται από ακόμα ένα άλλο αντικείμενο το οποίο ονομάζεται View. Το αντικείμενο View προσδιορίζει όλές τις παραμέτρους που χρειάζονται για να αποδώσει την σκηνή από το οπτικό πεδίο

του ViewPlatform. Τέλος υπάρχουν κάποια άλλα αντικείμενα όπου αναφέρονται στο View και περιέχουν πληροφορίες σχετικά με το καμβά σχεδίασης, την οθόνη που περιέχει τον καμβά και το φυσικό περιβάλλον.

Ο γράφος σκηνής του VirtualUniverse απεικονίζεται στο Σχήμα 16 και τα σύμβολα του γράφου σκηνής απεικονίζονται στο Σχήμα 17:

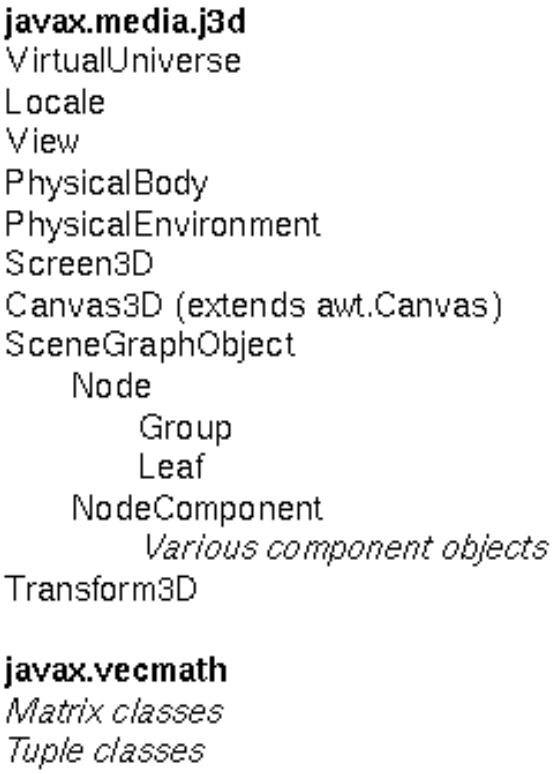


Σχήμα 16 "Ο γράφος σκηνής"



Σχήμα 17 "Τα σύμβολα του γράφου σκηνής"

Η Java 3D αποτελείται από ορισμένες βασικές κλάσεις οι οποίες χρησιμοποιούνται για την κατασκευή και το χειρισμό μιας σκηνής γράφου και για να ελέγχουν την οπτική εικόνα και την απόδοση. Στο Σχήμα 18 παρακάτω φαίνεται μια συνολική ιεραρχία των αντικειμένων που χρησιμοποιούνται στη Java 3D:



Σχήμα 18 "Οι κλάσεις της javax.media.j3d και της javax.vecmath"

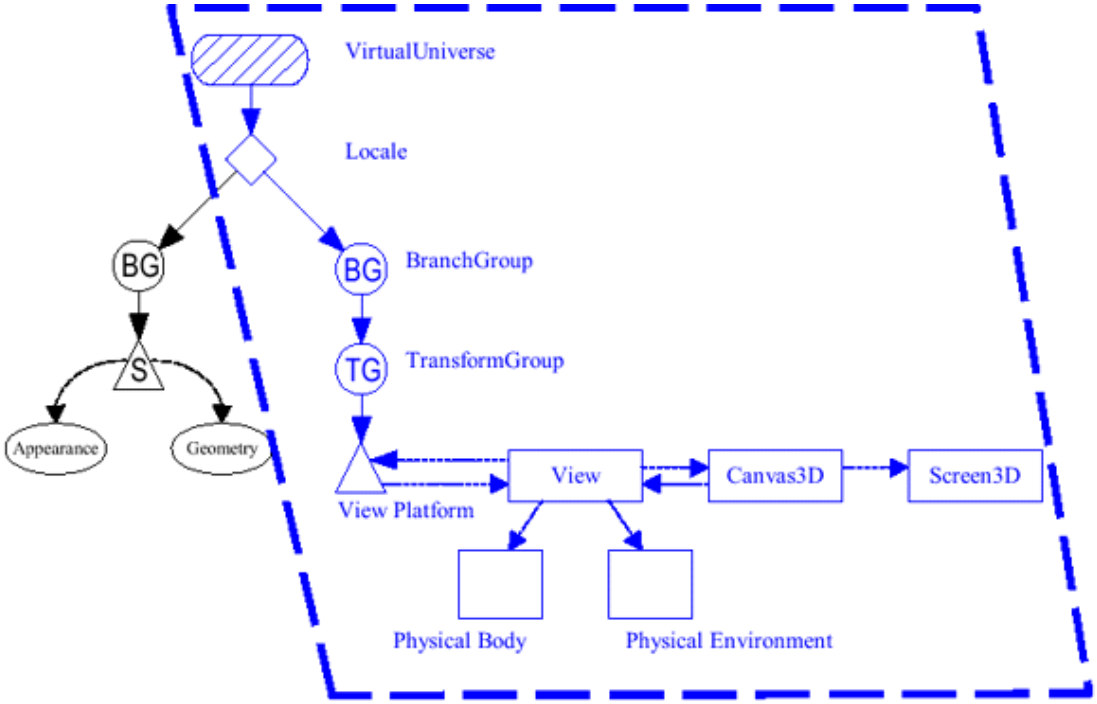
ΥΠΟΚΕΦΑΛΑΙΟ

6.4 Η ΚΛΑΣΗ SIMPLEUNIVERSE

Η κλάση SimpleUniverse χρησιμοποιείται για τη δημιουργία ενός γράφου σκηνής όπως και η VirtualUniverse. Ποια είναι η διαφορά μεταξύ αυτών των δύο κλάσεων; Η κλάση SimpleUniverse ιδρύει ένα μικρότερο περιβάλλον χρήστη έτσι ώστε ο προγραμματιστής να δημιουργεί και να εκτελεί εύκολα και γρήγορα ένα Java 3D πρόγραμμα. Η χρησιμότητα αυτής της κλάσης είναι ότι δημιουργεί όλα τα απαραίτητα αντικείμενα που έχουν να κάνουν με το οπτικό πεδίο του γράφου σκηνής που έχει ο χρήστης. Συγκεκριμένα αυτή η κλάση κατασκευάζει ένα locale, ένα μονό ViewingPlatform και ένα αντικείμενο Viewer με εξ ορισμού (default) τιμές. Σε πολλές βασικές Java 3D εφαρμογές θα ανακαλύψουμε ότι η SimpleUniverse παρέχει την απαραίτητη λειτουργικότητα που χρειάζεται να έχουν οι εφαρμογές

μας. Όμως υπάρχουν κάποιες πιο πολύπλοκες εφαρμογές που μπορεί να χρειάζονται περισσότερο έλεγχο έτσι ώστε να προσφέρουν επιπλέον λειτουργικότητα. Τέτοιου είδους εφαρμογές δεν είναι ικανές να χρησιμοποιήσουν την κλάση SimpleUniverse. Σε αυτές τις περιπτώσεις κάνουμε χρήση της VirtualUniverse που αναφερθήκαμε στην προηγούμενη παράγραφο.

Όπως μπορούμε να δούμε στο Σχήμα 19, η κλάση SimpleUniverse κατασκευάζει αυτόματα ένα πρότυπο View BranchGraph το οποίο απεικονίζεται μέσα στην μπλε διακεκομμένη γραμμή. Έτσι λοιπόν το μόνο που πρέπει να κάνουμε είναι να καθορίσουμε το περιεχόμενο του γράφου σκηνής:



Σχήμα 19 "Το διάγραμμα της κλάσης SimpleUniverse"

Η σειρά ενεργειών που γίνονται για την κατασκευή ενός SimpleUniverse είναι:

1. Δημιουργεί το αντικείμενο Canvas3D. Δημιουργεί το SimpleUniverse αντικειμένου.
2. Δημιουργεί το αντικείμενο SimpleUniverse το οποίο αναφέρεται στο Canvas3D.
3. Κατασκευάζει ένα BranchGroup και ορίζει το περιεχόμενο του.
4. Μεταγλωττίζει το περιεχόμενο του BranchGroup.
5. Προσκολλάει το περιεχόμενο του BranchGroup στο Locale του SimpleUniverse

ΥΠΟΚΕΦΑΛΑΙΟ
6.5 Η ΚΛΑΣΗ LOCALE

Ένα αντικείμενο Locale καθορίζει μία θέση υψηλής ανάλυσης μέσα στο VirtualUniverse, και χρησιμεύει ως container μιας συλλογής από BranchGroup υπογράφων. Τα αντικείμενα μέσα στο Locale ορίζονται δίνοντας τις ακριβείς συντεταγμένες που σχετίζονται με το σημείο δημιουργίας του Locale. Το σημείο δημιουργίας καθορίζει το σύστημα συντεταγμένων του ψηφιακού κόσμου για το Locale.

Η κατασκευή ενός αντικειμένου Locale Μπορεί να γίνει με τη χρήση τριών κατασκευαστών:

- Locale(VirtualUniverse universe) – Κατασκευάζει και αρχικοποιεί ένα υψηλής ανάλυσης Locale αντικείμενο στη θέση (0, 0, 0).
- Locale(VirtualUniverse universe, HiResCoord hiRes) - Κατασκευάζει και αρχικοποιεί ένα υψηλής ανάλυσης Locale αντικείμενο στη θέση που έχει οριστεί στο HiResCoord αντικείμενο {HiResCoord(int[] X, int[] Y, int[] Z)}.
- Locale(VirtualUniverse universe, int[] x, int[] y, int[] z) - Κατασκευάζει και αρχικοποιεί ένα υψηλής ανάλυσης Locale αντικείμενο στη θέση (x, y, z) που ορίζεται από τις παραμέτρους της μεθόδου.

Η κλάση Locale περιέχει μεθόδους για να ορίσουμε και να πάρουμε τις υψηλής ανάλυσης συντεταγμένες, καθώς και μεθόδους για να προσθέσουμε, να αφαιρέσουμε και να καταμετρήσουμε τους Branch graphs. Κάποιες από τις μεθόδους είναι:

- addBranchGraph(BranchGroup branchGroup) – Προσθέτει ένα νέο BranchGroup αντικείμενο στη λίστα των Branch Graphs
- getVirtualUniverse() – Επιστρέφει το VirtualUniverse που βρίσκεται το Locale.
- numBranchGraphs() – Επιστρέφει τον αριθμό των Branch Graphs στο Locale.
- removeBranchGraph(BranchGroup branchGroup) – Αφαιρεί το BranchGroup που ορίζουμε στην παράμετρο από την λίστα των Branch Graphs.
- replaceBranchGraph(BranchGroup oldGroup, BranchGroup newGroup) – Αντικαθιστά ένα ήδη υπάρχον BranchGroup με ένα καινούργιο BranchGroup στη λίστα των Branch Graphs.
- setHiRes(int[] x, int[] y, int[] z) – Ορίζει τις συνταγμένες (x, y, z) του Locale.

ΥΠΟΚΕΦΑΛΑΙΟ
6.6 Η ΚΛΑΣΗ BRANCHGROUP

Η κλάση BranchGroup χρησιμεύει ως δείκτης στη ρίζα του γράφου σκηνής. Τα αντικείμενα BranchGroup είναι τα μόνα αντικείμενα που μπορούν να εισαχθούν στο σύνολο αντικειμένων του Locale. Ένας υπογράφος, ο οποίος είναι ρίζα του BranchGroup, μπορεί να θεωρηθεί ως μεταγλωττισμένο αντικείμενο. Οι παρακάτω ενέργειες μπορούν να γίνουν με το BranchGroup:

- Ένα BranchGroup μπορεί να μεταγλωττιστεί απλά καλώντας την μέθοδο μεταγλώττισης. Αυτό προκαλεί μεταγλώττιση σε ολόκληρο τον υπογράφο. Αν υπάρχουν BranchGroup κόμβοι μέσα στον υπογράφο τότε θα μεταγλωττιστούν και αυτοί.
- Ένα BranchGroup μπορεί να μπει σε ένα Virtual Universe ή Simple Universe, προσκολλώντας αυτό στο Locale.
- Ένα BranchGroup το οποίο εμπεριέχεται μέσα σε ένα υπογράφο μπορεί να αλλάξει γονιό ή να αποκολληθεί κατά τη διάρκεια της εκτέλεσης εάν είναι δυνατόν αυτό να γίνει.

Επίσης είναι σημαντικό να σημειώσουμε ότι ένα BranchGroup περιλαμβάνει άλλον ένα υπογράφο σαν παιδί κάποιου άλλου κόμβου ομάδας, τότε δεν μπορεί να προσκολλήσει στο Locale.

Η δημιουργία ενός BranchGroup κόμβου γίνεται γράφοντας τον κώδικα:

```
BranchGroup objRoot = new BranchGroup();
```

Η κλάση BranchGroup διαθέτει ένα μεγάλο αριθμό από μεθόδους που βοηθάνε στη σωστή διαχείριση του περιεχομένου. Μερικές σημαντικές μέθοδοι είναι:

- compile() – Μεταγλωττίζει το BranchGroup που σχετίζεται με το αντικείμενο και δημιουργεί και αποθηκεύει το γράφο σκηνής.
- detach() – Αποκολλάει το BranchGroup από το γονιό του.
- addChild(Node παιδί) – Προσθέτει τον κόμβο παιδί, που ορίζουμε στην παράμετρο της μεθόδου, στη λίστα των κόμβων παιδιών. Εάν το ορισμένο αντικείμενο δεν είναι στη λίστα, τότε η λίστα δεν τροποποιείται.

- removeChild(Node child) – Αφαιρεί τον κόμβο παιδί, που ορίζουμε στην παράμετρο της μεθόδου, από τη λίστα των κόμβων παιδιών.
- removeAllChildren() – Αφαιρεί όλα τα παιδιά από τον κόμβο Group.
- setCapability(int bit) – Ορίζει την ικανότητα που θα έχει το BranchGroup. Μερικές από αυτές είναι:
    - ALLOW_CHILDREN_EXTEND: Επιτρέπει να προστίθενται νέα παιδιά στο Group κόμβο.
    - ALLOW_CHILDREN_READ: Επιτρέπει να διαβάζονται τα παιδιά στο Group κόμβο.
    - ALLOW_CHILDREN_WRITE: Επιτρέπει να γράφονται τα παιδιά στο Group κόμβο.
    - ALLOW_DETACH: Επιτρέπει στο BranchGroup να αποκολληθεί από το γονιό του.

ΥΠΟΚΕΦΑΛΑΙΟ
6.7 Η ΚΛΑΣΗ TRANSFORMGROUP

Η TransformGroup είναι ο κόμβος που σχετίζεται με το μετασχηματισμό του αντικειμένου μας. Ο κόμβος TransformGroup ουσιαστικά καθορίζει το μετασχηματισμό ενός σχήματος/αντικειμένου στο χώρο μας μέσω του αντικειμένου Transform3D. Ο μετασχηματισμός που μπορούμε να κάνουμε σε ένα αντικείμενο είναι: να το τοποθετήσουμε σε διαφορετικό σημείο στο χώρο, να το μεγεθύνουμε ή σμικρύνουμε, να το περιστρέψουμε ή να αλλάξουμε τον προσανατολισμό του.

Ο προσδιορισμένος μετασχηματισμός πρέπει να είναι σχετικός. Συγκεκριμένα, αν ο κόμβος TransformGroup χρησιμοποιείται σαν πρόγονος του κόμβου ViewPlatform στο γράφο σκηνής, τότε οι μετασχηματισμοί που είναι μόνο κατάλληλοι είναι: η περιστροφή (rotation), η επανατοποθέτηση (translation) και η διαβάθμιση (scaling). Και οι τρεις μετασχηματισμοί επιτρέπονται στο απευθείας μονοπάτι από το Locale στο ViewPlatform κόμβο. Ωστόσο, σε περίπτωση που επιτρέπονται αυθαίρετοι μετασχηματισμοί, θα είχαμε καλύτερη απόδοση εάν όλοι οι πίνακες (matrixes) μέσα στο Branch Graph ήταν κατάλληλοι, περιέχοντας μόνο περιστροφές, επανατοποθετήσεις και διαβαθμίσεις.

Οι δύο κατασκευαστές της TransformGroup είναι:

- TransformGroup() – Δημιουργεί και αρχικοποιεί ένα TransformGroup.
- TransformGroup(Transform3D t1) - Δημιουργεί και αρχικοποιεί ένα TransformGroup με βάση τους μετασχηματισμού της παραμέτρου.

Επίσης η κλάση TransformGroup έχει την μέθοδο setTransform(Transform3D t1) και τη μέθοδο getTransform(Transform3D t1) για να ορίζει και να παίρνει το μετασχηματισμό ενός αντικειμένου αντίστοιχα.

ΥΠΟΚΕΦΑΛΑΙΟ
6.8 Η ΚΛΑΣΗ TRANSFORM3D

Όπως αναφερθήκαμε και στην προηγούμενη παράγραφο, η κλάση Transform3D καθορίζει το μέγεθος, τη θέση και το προσανατολισμό του αντικειμένου στο χώρο. Ένα αντικείμενο Transform3D απεικονίζει ένα double ή float 4x4 πίνακα σημείων.

Ο τύπος του Transform3D υπολογίζεται εσωτερικά όταν το Tranform αντικείμενο κατασκευάζεται ή όταν ενημερώνεται για κάποια τροποποίηση. Ο τύπος του Transform3D ουσιαστικά καθορίζει τις τιμές που θα περιλαμβάνει ο πίνακας του αντικειμένου. Ένας πίνακας μπορεί να έχει πολλαπλούς τύπους. Οι διαθέσιμοι τύποι πίνακα είναι:

- ZERO: Όλες οι τιμές του πίνακα είναι 0.
- IDENTITY: Οι τιμές της διαγωνίου του πίνακα είναι 1 και οι υπόλοιπες 0.
- SCALE: Ο πίνακας είναι ένας ομοιόμορφος διαβαθμισμένος πίνακα, χωρίς συστατικά περιστροφής και επανατοποθέτησης.
- ORTHOGONAL: Η τιμές της διαβάθμισης είναι 1 και δεν υπάρχει το συστατικό της επανατοποθέτησης.
- RIGID: Η τιμές της διαβάθμισης είναι 1 και υπάρχει το συστατικό της επανατοποθέτησης.
- CONGRUENT: Μπορεί να υπάρξει επανατοποθέτηση, περιστροφή, αντανάκλαση και διαβάθμιση του αντικειμένου.

- AFFINE: Ένας συσχετιζόμενος πίνακας μπορεί να επανατοποθετήσει, περιστρέψει, αντανακλάσει, διαβαθμίσει ανισοτροπικά και να περικόψει.
- NEGATIVE_DETERMINANT: Ο πίνακας αυτό έχει αρνητική ορίζουσα.

Η Transform3D έχει ένα μεγάλο αριθμό κατασκευαστών. Μερικοί από αυτούς είναι:

- Transform3D() – Δημιουργεί και αρχικοποιεί ένα αντικείμενο Transform3D με πίνακα τύπου Identity.
- Transform3D(double[] matrix) - Δημιουργεί και αρχικοποιεί ένα αντικείμενο Transform3D με βάση τον 16 μήκους πίνακα της παραμέτρου.
- Transform3D(Matrix4d m1) - Δημιουργεί και αρχικοποιεί ένα αντικείμενο Transform3D με βάση τον 4x4 πίνακα της παραμέτρου.
- Transform3D(Transform3D t1) - Δημιουργεί και αρχικοποιεί ένα αντικείμενο Transform3D με βάση το αντικείμενο Transform3D της παραμέτρου.

Η κλάση Transform3D διαθέτει ένα μεγάλο αριθμό μεθόδων προσφέροντας στο προγραμματιστή ευελιξία στο μετασχηματισμό των αντικειμένων του. Οι σημαντικότερες μέθοδοι της Transform3D είναι:

- rotX/rotY/rotZ(double angle) – Περιστρέφει αριστερόστροφα το αντικείμενο στον άξονα x, y, z αντίστοιχα βάσει τη γωνία της παραμέτρου.
- setScale(double scale) – Ορίζει το μέγεθος του αντικείμενου.
- mul(Transform3D t1) – Πολλαπλασιάζει τις τιμές αυτού του Transform3D με τις τιμές του t1 Transform3D (this = this*t1).
- setTranslation(Vector3d trans) – Ορίζει τη νέα τοποθεσία του αντικειμένου μέσω του διανύσματος trans. Το διάνυσμα αυτό περιέχει τις τιμές x, y και z που καθορίζουν το σημείο στο χώρο.
- add(Transform3D t1) - Προσθέτει τις τιμές αυτού του Transform3D με τις τιμές του t1 Transform3D (this = this+t1).
- getType() – Επιστρέφει τον τύπο του πίνακα ως ένα bitmask που περιγράφει σε ποιο τύπο ανήκει.
- sub(Transform3D t1) - Αφαιρεί τις τιμές αυτού του Transform3D με τις τιμές του t1 Transform3D (this = this-t1).

ΥΠΟΚΕΦΑΛΑΙΟ
6.9 ΔΗΜΙΟΥΡΓΙΑ ΓΕΩΜΕΤΡΙΚΩΝ ΣΧΗΜΑΤΩΝ

Η Java 3D παρέχει τέσσερεις κλάσεις για την κατασκευή βασικών γεωμετρικών σχημάτων. Οι κλάσεις αυτές είναι η Box, η Cone, η Cylinder και η Sphere, οι οποίες είναι υποκλάσεις του com.sun.j3d.utils.geometry.Primitive. Επίσης τα αντικείμενα αυτών των σχημάτων προστίθενται ως κόμβος παιδί στο αντικείμενο TransformGroup το οποίο σχετίζεται με τη μορφή του αντικειμένου στο χώρο. Για να δημιουργήσουμε ένα γεωμετρικό σχήμα θα πρέπει να δώσουμε τις διαστάσεις του, π.χ. μήκος, πλάτος, ύψος, κ.λπ. Οι βασικές μέθοδοι του κάθε σχήματος είναι:

- Box() – Κατασκευάζει ένα κουτί ορίζοντας την τιμή 1,0 σε όλες τις διαστάσεις.
- Box(float xdim, float ydim, float zdim, Appearance ap) - Κατασκευάζει ένα κουτί με τις διαστάσεις και την εμφάνιση που ορίζονται στις παραμέτρους.
- Cone() – Κατασκευάζει ένα κώνο με ακτίνα 1,0 και ύψος 2,0.
- Cone(float radius, float height, Appearance ap) - Κατασκευάζει ένα κώνο με τις διαστάσεις και την εμφάνιση που ορίζονται στις παραμέτρους.
- Cylinder() - Κατασκευάζει ένα κύλινδρο με ακτίνα 1,0 και ύψος 2,0.
- Cylinder(float radius, float height, Appearance ap) - Κατασκευάζει ένα κύλινδρο με τις διαστάσεις και την εμφάνιση που ορίζονται στις παραμέτρους.
- Sphere() - Κατασκευάζει μία σφαίρα με ακτίνα 1,0.
- Sphere(float radius, Appearance ap) - Κατασκευάζει μία σφαίρα με την ακτίνα και την εμφάνιση που ορίζονται στις παραμέτρους.

Η κλάση Appearance σχετίζεται με την εμφάνιση του γεωμετρικού σχήματος στο χώρο. Περιλαμβάνει χαρακτηριστικά όπως το χρώμα, το υλικό, την υφή, τη διαφάνεια του σχήματος κ.α. Για το χειρισμό αυτών των χαρακτηριστικών η Appearance διαθέτει αρκετές μεθόδους. Κάποιες από αυτές είναι:

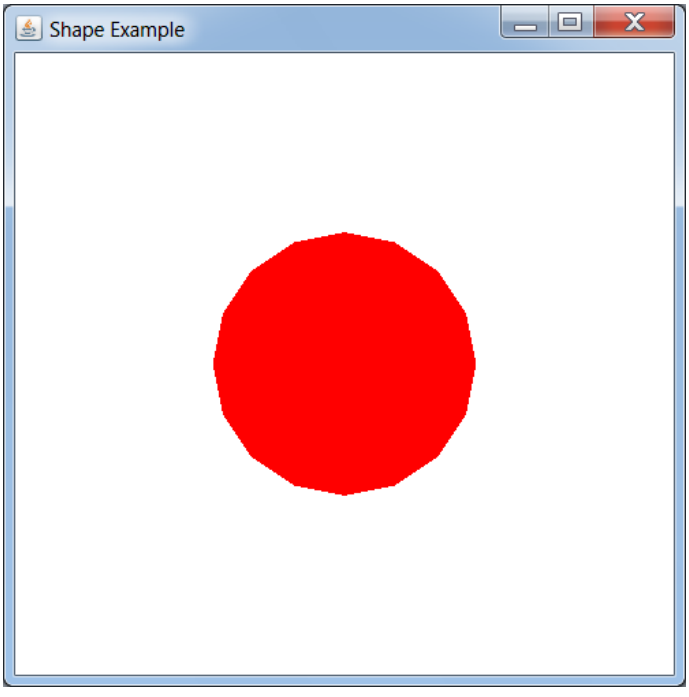
- setColoringAttributes(ColoringAttributes coloringAttributes) – Ορίζει το χρώμα που θα έχει το αντικείμενο.
- setMaterial(Material material) – Ορίζει το υλικό που θα έχει το αντικείμενο.
- setTexture(Texture texture) - Ορίζει την υφή που θα έχει το αντικείμενο.

- setTransparencyAttributes(TransparencyAttributes transparencyAttributes) – Ορίζει τη διαφάνεια που θα έχει το αντικείμενο.

Στο παρακάτω παράδειγμα που ακολουθεί, δημιουργείται ένας γράφος σκηνής με τη χρήση της κλάσης SimpleUniverse. Μέσα σε αυτό το χώρο κατασκευάζεται μία κόκκινη σφαίρα (Σχήμα 20). Για να έχει κόκκινο χρώμα η σφαίρα μας φτιάχνουμε ένα αντικείμενο Appearance και του ορίζουμε το χρώμα.

```
import java.applet.*;
import java.awt.BorderLayout;
import java.awt.GraphicsConfiguration;
import com.sun.j3d.utils.universe.*;
import com.sun.j3d.utils.geometry.Sphere;
import java.awt.Color;
import javax.media.j3d.*;
import javax.swing.JFrame;
import javax.swing.JPopupMenu;
import javax.swing.ToolTipManager;
import javax.vecmath.*;

public class ShapeExample extends Applet {
    public ShapeExample() {
        //Δημιουργεί μία διάταξη γραφικών για SimpleUniverse
        GraphicsConfiguration config =
SimpleUniverse.getPreferredConfiguration();
        setLayout(new BorderLayout());

        Canvas3D canvas3D = new Canvas3D(config);  //Δημιουργία του καμβά
        add("Center", canvas3D);
        //Κάλεσμα της μεθόδου scene για τη κατασκευή της σφαίρας
        BranchGroup scene = createSceneGraph();

//Δημιουργία του SimpleUniverse και προσθήκη του BranchGroup σε αυτό
        SimpleUniverse simpleU = new SimpleUniverse(canvas3D);
        simpleU.getViewingPlatform().setNominalViewingTransform();
        simpleU.addBranchGraph(scene);
    }

    public BranchGroup createSceneGraph() {
        BranchGroup objRoot = new BranchGroup();  //Δημιουργία του Bgroup
        //Δημιουργία του Appearance και καθορισμός του χρώματος
```

```
        Appearance ap = new Appearance();
        ColoringAttributes ca = new ColoringAttributes();
        ca.setColor(new Color3f(Color.RED));
        ap.setColoringAttributes(ca);

        Sphere sp = new Sphere(0.4f, ap); //Δημιουργία της σφαίρας και
        objRoot.addChild(sp);             //προσθήκη στο BranchGroup

        //Δημιουργία ενός άσπρου φόντου και προσθήκη στο BranchGroup
        Background background = new Background(new Color3f(Color.WHITE));
        background.setApplicationBounds(new            BoundingSphere(new
Point3d(0.0, 0.0, 0.0), 100.0));
        objRoot.addChild(background);
        return objRoot;
    }
    public static void main(String[] args) {
        //Κατασκευή του πλαισίου JFrame
        JFrame frame = new JFrame("Shape Example");
        ShapeExample shapeEx = new ShapeExample();
        frame.getContentPane().add(shapeEx);
        frame.setSize(500, 500);
        frame.setVisible(true);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
}
```



Σχήμα 20 "Το αποτέλεσμα της εκτέλεσης του ShapeExample"

ΥΠΟΚΕΦΑΛΑΙΟ

6.10 ΠΡΟΣΘΗΚΗ ΦΩΤΙΣΜΟΥ ΣΤΟ ΓΡΑΦΟ ΣΚΗΝΗΣ

Η Java 3D API διαθέτει την αφηρημένη κλάση Light η οποία χειρίζεται το φωτισμό στο γράφο σκηνής. Η κλάση Light είναι μια κλάση φύλλο (Leaf) και διαθέτει ένα σύνολο παραμέτρων για τη σωστή διαχείριση όλων των τύπων φωτισμού. Οι παράμετροι αυτοί περιλαμβάνουν το χρώμα φωτός, τις επιτρεπόμενες σημαίες και την περιοχή που θα επηρεάζει το φως. Επίσης ο κόμβος Light περιέχει μία λίστα με Groups, στην οποία δηλώνεται το ιεραρχικό φάσμα του φωτός. Αν η λίστα φάσματος είναι άδεια, τότε ο κόμβος Light έχει γενικό φάσμα, δηλαδή όλοι η κόμβοι που είναι στην περιοχή του φωτός επηρεάζονται από αυτό.

Το φως στο γράφο σκηνής μπορεί να προέρχεται από διαφορετικές πηγές οι οποίες μπορούν να ορίζονται ξεχωριστά. Κάποιο φως της σκηνής μπορεί να προέρχεται από μια συγκεκριμένη κατεύθυνση, γνωστό ως κατευθυνόμενο φως, ή από μια συγκεκριμένη θέση, γνωστό ως φως σημείου, ή από μία μη ιδιαίτερη κατεύθυνση ή πηγή, γνωστό ως φως περιβάλλοντος. Αυτά τα τρία διαφορετικά είδη φωτός υλοποιούνται στις υποκλάσεις της Light: AmbientLight, DirectionalLight και PointLight.

Οι δύο σημαντικοί κατασκευαστές της κλάσης Light είναι:

- Light(boolean lightOn, Color3f color) – Δημιουργεί ένα κόμβο Light και ορίζει το χρώμα του φωτός και αν θα είναι ενεργό ή όχι.
- Light(Color3f color) – Κατασκευάζει ένα Light και ορίζει το χρώμα του φωτός.

Η κλάση Light έχει αρκετές μεθόδους για το χειρισμό του φωτός. Μερικές είναι:

- addScope(Group scope) – Προσθέτει το Group της παράμετρο στη λίστα φάσματος του Light.
- setColor(Color3f color) – Ορίζει το χρώμα του φωτός.
- setEnable(boolean state) – Ορίζει αν θα είναι ενεργό το φως η όχι.
- setInfluencingBounds(Bounds region) – Ορίζει την περιοχή που θα επηρεάζει το φως.
- setScope(Group scope, int index) – Αντικαθιστά το Group που βρίσκεται προσδιορισμένο δείκτη της λίστας φάσματος με το Group της παραμέτρου.

ΥΠΟΚΕΦΑΛΑΙΟ

6.10.1 Η κλάση AmbientLight

Όταν θέλουμε στο γράφο σκηνής μας ένα αντικείμενο περιβάλλοντος ως πηγή φωτός, τότε χρησιμοποιούμε την κλάση AmbientLight. Το φως περιβάλλοντος είναι ένα φως που προέρχεται από όλες τις κατευθύνσεις. Το φως περιβάλλοντος έχει τα ίδια χαρακτηριστικά με την κλάση Light με τη διαφορά ότι οι ανακλάσεις δεν εξαρτώνται από τον προσανατολισμό ή τη θέση της επιφάνειας. Το φως περιβάλλοντος έχει μόνο ένα συστατικό ανάκλασης. Η κατασκευαστές της AmbientLight παίρνουν τις ίδιες παραμέτρους με τη Light: AmbientLight(Color3f color) και AmbientLight(boolean lightOn, Color3f color).

Η μόνη μέθοδος που διαθέτει η AmbientLight είναι: cloneNode(boolean forceDuplicate) με τη οποία δημιουργούμε νέο παράδειγμα του κόμβου. Επιπλέον κληρονομεί όλες τις μεθόδους της κλάσης Light.

ΥΠΟΚΕΦΑΛΑΙΟ

6.10.2 Η κλάση DirectionalLight

Ένας κόμβος DirectionalLight ορίζει τον προσανατολισμό ενός φωτός με αφετηρία υπόστασης το άπειρο. Η DirectionalLight έχει τα ίδια χαρακτηριστικά με την κλάση Light, καθώς και ένα κατευθυντήριο διάνυσμα για τον καθορισμό της κατεύθυνσης όπου το φως λάμπει. Ένα κατευθυνόμενο φως έχει παράλληλες ακτίνες οι οποίες ταξιδεύουν σε μία κατεύθυνση κατά μήκος του διανύσματος. Επίσης μπορεί να διαχυθεί και να ανακλαστεί, αλλά όμως δεν μπορεί να προσφέρει ανάκλαση περιοχής. Οι κατασκευαστές της DirectionalLight παίρνουν μία επιπλέον παράμετρο την Vector3f direction η οποία ορίζει το διάνυσμα της κατεύθυνσης που θα έχει φως: DirectionalLight(boolean lightOn, Color3f color, Vector3f direction) και DirectionalLight(Color3f color, Vector3f direction).

Η DirectionalLight περιλαμβάνει δύο νέες μεθόδους πέραν από αυτών που κληρονομεί από την κλάση Light:

- getDirection(Vector3f direction) – Επιστρέφει το διάνυσμα που δηλώνει την κατεύθυνση του φωτός.
- setDirection(Vector3f direction) – Ορίζει το διάνυσμα της κατεύθυνσης που θα έχει το φως.

ΥΠΟΚΕΦΑΛΑΙΟ

6.10.3 Η κλάση PointLight

Ένα αντικείμενο PointLight προσδιορίζει μία εξασθενημένη πηγή φωτός σε ένα σταθερό σημείο στο χώρο, το οποίο ακτινοβολεί ισομερώς φως σε όλες τις κατευθύνσεις μακριά από την πηγή. Η PointLight έχει τα ίδια χαρακτηριστικά με την Light, καθώς επίσης τις παραμέτρους της τοποθεσίας και την εξασθένηση του φωτός. Όπως και το DirectionLight το φως σημείου μπορεί να διαχυθεί και να ανακλαστεί, αλλά όμως δεν μπορεί να προσφέρει ανάκλαση περιοχής. Ο συντελεστής της εξασθένησης του φωτός προκύπτει από τρεις τιμές: την μόνιμη, την γραμμική και την τετραγωνική εξασθένηση. Οι κατασκευαστές της PointLight έχουν της ίδιες παραμέτρους με αυτούς της Light, με τη διαφορά ότι παίρνουν δύο επιπλέον παραμέτρους: την Point3f position η οποία δηλώνει το σημείο της πηγής του φωτός και την Point3f attenuation η οποία δηλώνει την εξασθένηση του φωτός.

- PointLight(boolean lightOn, Color3f color, Point3f position, Point3f attenuation)
- PointLight(Color3f color, Point3f position, Point3f attenuation)

Δύο αρκετά σημαντικές μέθοδοι της κλάσης PointLight είναι:

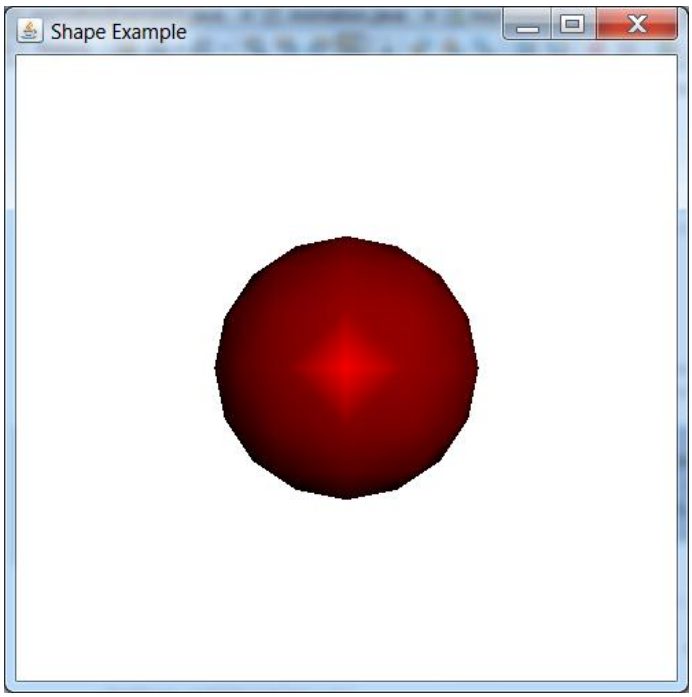
- setAttenuation(float constant, float linear, float quadratic) – Ορίζει τις τρεις τιμές με τις οποίες υπολογίζεται ο συντελεστής εξασθένησης.
- setPosition(Point3f position) – Ορίζει τη θέση του φωτός.

ΥΠΟΚΕΦΑΛΑΙΟ

6.10.4 Δημιουργία κατευθυντήριου φωτισμού στο ShapeExample

Αρχικά για να μπορέσουμε να δούμε το φως που ανακλάται πάνω στη σφαίρα, θα πρέπει να αλλάξουμε το χρώμα της σφαίρας από κόκκινο σε λευκό. Έπειτα όπως είπαμε στις προηγούμενες παραγράφους πρέπει να δηλώσουμε την περιοχή που θα επηρεάζει το φώς, το χρώμα του και επειδή είναι DirectionLight την κατεύθυνση που θα έχει. Έτσι λοιπόν θα προσθέσουμε στην μέθοδο createSceneGraph() τις παρακάτω γραμμές κώδικα για να δημιουργήσουμε φωτισμό στο γράφο σκηνή (Σχήμα 21) και να μετατρέψουμε τα γραφικά πιο όμορφα και ρεαλιστικά στο χρήστη:

```
Color3f lightColor = new Color3f(Color.RED);
BoundingSphere bounds = new BoundingSphere(new Point3d(0.0, 0.0, 0.0),
100.0);
Vector3f lightDirection = new Vector3f(-1.0f, -1.0f, -5.0f);
DirectionalLight dl = new DirectionalLight(lightColor, lightDirection);
dl.setInfluencingBounds(bounds);
objRoot.addChild(dl);
```



Σχήμα 21 "Χρήση DirectionalLight στο ShapeExample"

ΥΠΟΚΕΦΑΛΑΙΟ
6.11 Η ΚΛΑΣΗ VIEW

Ένα αντικείμενο View περιέχει όλες τις παραμέτρους που απαιτούνται για να γίνει μία σκηνή τριών διαστάσεων από μία οπτική γωνία. Η κλάση view διαθέτει μία λίστα αντικειμένων της Canvas3D, την οποία η view διαμορφώνει. Αυτή υπάρχει έξω από το γράφο σκηνής, αλλά συνδέεται με ένα αντικείμενο ViewPlatform κόμβο στο γράφο σκηνής. Επίσης αυτή περιέχει μία αναφορά στο PhysicalBody και στο PhysicalEnvironment αντικείμενο.

Το αντικείμενο View είναι ένα βασικό Java 3D αντικείμενο για τον έλεγχο του Java 3D μοντέλου προβολής. Όλα τα συστατικά τα οποία καθορίζουν τον μετασχηματισμό της View και χρησιμοποιούνται για να αποδοθούν στου 3D καμβάδες είτε εμπεριέχονται στο αντικείμενο View είτε είναι αντικείμενα που αναφέρονται στο αντικείμενο View. Έτσι λοιπόν η Java 3D επιτρέπει τις εφαρμογές

να καθορίζουν πολλαπλά ενεργά αντικείμενα View ταυτόχρονα, όπου το κάθε ένα ελέγχει το δικό του σύνολο καμβάδων.

Η κλάση View έχει πολλές μεταβλητές και μεθόδους, όμως οι περισσότερες είναι μεταβλητές βαθμονόμησης ή βοηθητικές λειτουργίες χρήστη. Κάποιες σημαντικές μέθοδοι είναι:

- addCanvas3D(Canvas3D canvas3D) – Προσθέτει τον καμβά στη λίστα.
- attachViewPlatform(ViewPlatform vp) – Συνδέει το αντικείμενο ViewPlaform με το View.
- removeCanvas3D(Canvas3D canvas3D) – Αφαιρεί τον καθορισμένο canva3D από τη Canvas3D λίστα της View.
- repaint() – Αιτείται να αποτυπωθεί το View όσο το δυνατόν συντομότερα.
- setFrontClipDistance(double distance) – Ορίζει την μπροστινή απόσταση του μοντέλου View.
- setPhysicalBody(PhysicalBody physicalBody) – Ορίζει το PhisicalBody του μοντέλου View.
- setPhysicalEnvironment(PhysicalEnvironment physicalEnvironment) - Ορίζει το PhisicalEnvironment του μοντέλου View.

ΥΠΟΚΕΦΑΛΑΙΟ

6.11.1 Η κλάση PhysicalBody

Το αντικείμενο PhysicalBody περιέχει την περιγραφή του κεφαλιού του χρήστη. Οι ιδιότητες αυτού του αντικειμένου ορίζονται με το σύστημα συντεταγμένων του κεφαλιού. Η αφετηρία ορίζεται να είναι η μέση διαδρομή ανάμεσα στο αριστερό και το δεξί μάτι στο πρόσωπο του χαρακτήρα. Ο άξονας x επεκτείνεται προς τα δεξιά, ο άξονας y επεκτείνεται προς τα πάνω και ο άξονας z επεκτείνεται προς το πίσω μέρος του κεφαλιού. Έτσι λοιπόν μπορούμε εύκολα να ορίσουμε τα σημεία του αριστερού και του δεξιού ματιού στο κεφάλι. Πέραν από τις θέσεις των ματιών μπορούμε να δηλώσουμε και τις θέσεις των αυτιών στο κεφάλι. Οι δύο χρήσιμοι κατασκευαστές της PhysicalBody είναι: PhysicalBody(Point3d leftEyePosition, Point3d rightEyePosition) και PhysicalBody(Point3d leftEyePosition, Point3d rightEyePosition, Point3d leftEarPosition, Point3d rightEarPosition).

Επιπλέον η κλάση PhysicalBody περιέχει μεθόδους για τον ορισμό και τη επιστροφή των θέσεων των ματιών και των αυτιών. Μερικές από αυτές είναι:

- getRightEarPosition(Point3d position)
- getRightEyePosition(Point3d position)
- setLeftEarPosition(Point3d position)
- setLeftEyePosition(Point3d position)

ΥΠΟΚΕΦΑΛΑΙΟ

6.11.2 Η κλάση PhysicalEnvironment

Το αντικείμενο PhysicalEnvironment περιέχει την περιγραφή του φυσικού περιβάλλοντος στο οποίο η view θα παραχθεί. Αυτό χρησιμοποιείται για να δημιουργήσουμε συσκευές εισόδου (αισθητήρες) για τη παρακολούθηση του κεφαλιού και άλλες χρήσεις, καθώς και συσκευή εξόχου ήχου. Οι αισθητήρες προσαρμόζονται ξεκινώντας από το μηδέν. Η κλάση Sensor (αισθητήρας) περιλαμβάνει ένα αντικείμενο το οποίο παρέχει δεδομένα πραγματικού χρόνου. Για παράδειγμα ένα joystick ή ένα αρχείο δεδομένων του διαβάζεται κατά τη διάρκεια ενός προγράμματος. Ένας αισθητήρας, πρέπει να χρησιμοποιηθεί σε συνδυασμό με μια εφαρμογή της διεπαφής InputDevice. Οι δύο κατασκευαστές της PhysicalEnvironment: PhysicalEnvironment() και PhysicalEnvironment(int sensorCount) όπου sensorCount ο αριθμός των αισθητήρων που θα χρησιμοποιήσουμε.

Οι σημαντικότερες μέθοδοι της PhysicalEnvironment είναι:

- addInputDevice(InputDevice device) – Προσθέτει μία συσκευή εισόδου στη λίστα των συσκευών εισόδου.
- setAudioDevice(AudioDevice device) – Ορίζει το αντικείμενο AudioDevice ως τη συσκευή με την οποία θα παράγεται ήχος.
- setHeadIndex(int index) – Ρυθμίζει το δείκτη του κεφαλιού στο συγκεκριμένο δείκτη αισθητήρα
- setLeftHandIndex(int index) - Ρυθμίζει το δείκτη του αριστερού χεριού στο συγκεκριμένο δείκτη αισθητήρα.
- setLeftHandIndex(int index) - Ρυθμίζει το δείκτη του δεξιού χεριού στο συγκεκριμένο δείκτη αισθητήρα.

ΕΠΙΛΟΓΟΣ

Στο κεφάλαιο αυτό μας δόθηκε η ευκαιρία να εστιάσουμε σε ένα μεγάλο και απαιτητικό αντικείμενο της Java, αυτό των τρισδιάστατων γραφικών (Java 3D). Έτσι λοιπόν αναλύσαμε εκτενέστερα την ιεραρχία του γράφου σκηνής και τη διαφορά ανάμεσα στο VirtualUniverse και SimpleUniverse. Τέλος περιγράψαμε αναλυτικά τις σημαντικές Java 3D κλάσεις και μεθόδους, καθώς και πως συνδέονται μεταξύ τους έτσι ώστε να δημιουργήσουμε ένα πρόγραμμα γραφικών.

ΣΥΜΠΕΡΑΣΜΑΤΑ

Η πτυχιακή αυτή εργασία είχε ως σκοπό τη διερεύνηση των δυνατοτήτων της Java, που έχουν να κάνουν με την ανάπτυξη παιχνιδιών, τρισδιάστατων γραφικών και animation. Η Java παρέχει στους χρήστες της ένα μεγάλο αριθμό βιβλιοθηκών που τους βοηθάει στην κατανόηση της λειτουργία της και στον εύκολο προγραμματισμό. Αφού καταγράφηκε και παρουσιάστηκε η χρήση και η εφαρμογή της κάθε βιβλιοθήκης, η εργασία παρουσίασε μέσω των τριών διαφορετικών εφαρμογών (SpaceInvaders, Graphics3D και Animation) κάποιες από τις πραγματικά πολλές δυνατότητες της Java. Η ανάπτυξη αυτών των προγραμμάτων έγινε έτσι ώστε να υπάρξει εξοικείωση με το σχεδιασμό τέτοιων ειδικών προγραμμάτων και να διευρυνθεί η γνώση στη Java. Επίσης σκοπός της εργασίας δεν ήταν η χρησιμοποίηση βέλτιστων αλγορίθμων γραφικών, καθώς και σχεδίαση πλοκής παιχνιδιών. Αντίθετα χρησιμοποιώντας τις έτοιμες βιβλιοθήκες της Java και συνδέοντας όλες αυτές με τον κατάλληλο κώδικα, είχαμε την ευκαιρία να πλησιάσουμε αρκετά κοντά σε ένα μέχρι πρότινος κλειστό και απαιτητικό αντικείμενο όπως αυτό του προγραμματισμού παιχνιδιών, τρισδιάστατων γραφικών και animation.

Κάθε χρόνο η τεχνολογία αναπτύσσεται με ραγδαίους ρυθμούς με αποτέλεσμα να επηρεάζει την ανθρωπότητα. Στις μέρες μας όλοι οι υπολογιστές, κινητά και άλλα συστήματα, που διαθέτουν οι άνθρωποι, έχουν τη δυνατότητα να τρέχουν εφαρμογές όπως παιχνίδια, πολυμέσα, animation κ.λπ. Η ανάγκη προγραμματιστών ανάπτυξης τέτοιων ειδών εφαρμογών θα αυξάνεται όλο και περισσότερο κάθε χρόνο. Έτσι λοιπόν θεωρείται απαραίτητη η στοιχειώδης γνώση της ανάπτυξης παιχνιδιών, γραφικών και animation ενός προγραμματιστή Java, εάν θέλει να εστιάσει στο χώρο των γραφικών. Τέλος μην ξεχνάμε ότι τα τελευταία χρόνια η Sun έχει δείξει μεγαλύτερη αφοσίωση στην ανάπτυξη παιχνιδιών.

ΒΙΒΛΙΟΓΡΑΦΙΑ

Liakeas, G. (2009). Introduction to Java, Kleidarithmos

Brackeen, D. et al. (2004). Developing Games in Java, New Riders Publishing

Cinnamon, I. (2008). Programming Video Games for the Evil Genius, McGraw-Hill

Davison, A. (2005). Killer Game Programming in Java, O'Reilly

Harbour, S. J. (2012). Beginning Java SE 6 Game Programming, Third Edition, Course Technology

ΟΔΗΓΟΣ ΧΡΗΣΗΣ ΛΟΓΙΣΜΙΚΟΥ

Στο πρακτικό μέρος της πτυχιακής εργασίας ζητήθηκε η ανάπτυξη ενός απλού παιχνιδιού, ενός προγράμματος γραφικών και μιας σύντομης animation ταινίας με Java. Σκοπός αυτών των προγραμμάτων είναι η επίδειξη και η κατανόηση όλων των βιβλιοθηκών και μεθόδων που περιγράφει αυτή η πτυχιακή εργασία στα κεφάλαια της. Το λογισμικό που χρησιμοποιήθηκε για τη δημιουργία των προγραμμάτων ήταν το NetBeans IDE 7.1 και το CorelDRAW 12.

ΤΟ ΠΡΟΓΡΑΜΜΑ SPACE INVADERS

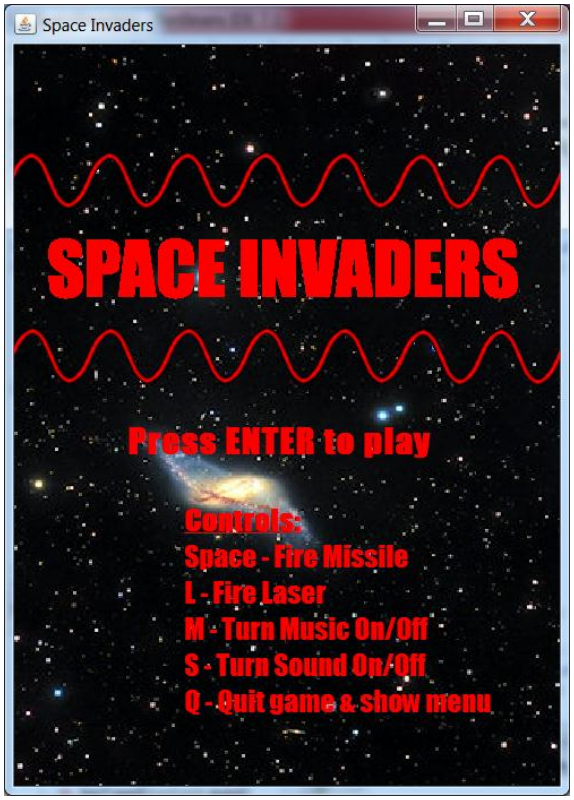
Το πρόγραμμα Space Invaders είναι ένα Spaceship Shooter παιχνίδι δύο διαστάσεων. Γενικά στην κατηγορία παιχνιδιών Spaceship Shooter ο παίκτης έχει τη δυνατότητα να πιλοτάρει ένα αεροσκάφος ή διαστημόπλοιο και με τα όπλα που έχει στη διάθεση του να καταρρίπτει τα εχθρικά αεροσκάφη. Επίσης θα πρέπει να είναι πολύ προσεκτικός και να αποφεύγει τα εχθρικά πυρά, έτσι ώστε να μην αποτύχει και τερματίσει το παιχνίδι. Στο Space Invaders γίνεται ακριβώς ότι περιγράψαμε με τη μόνη διαφορά ότι οι εχθροί αναπαράγονται με αυξητικούς ρυθμούς κάθε φορά που καταρρίπτουμε τους εχθρούς και ότι το παιχνίδι δεν τερματίζει ποτέ. Ουσιαστικά ο παίκτης κάθε φορά που καταστρέφει ένα εχθρό ή παίρνει τα ειδικά κιβώτια κερδίζει πόντους με τους οποίους στο τέλος σημειώνεται η τελική του βαθμολογία. Η βαθμολογία καθορίζει πόσο καλά τα πήγε ο παίκτης στο παιχνίδι.

Η πρώτη εικόνα που βλέπει ο χρήστης όταν εκτελεί το Space Invaders είναι το κύριο μενού, όπου εκεί του δίνεται η δυνατότητα πιέζοντας το πλήκτρο ENTER να ξεκινήσει να παίζει. Επίσης στο κύριο μενού εμφανίζεται ο χειρισμός του αεροσκάφους και του παιχνιδιού (Σχήμα 22).

Κατά τη διάρκεια της μάχης ο παίκτης μπορεί να δει στο κάτω μέρος του πλαισίου του παιχνιδιού το επίπεδο της πίστας, το σκορ και την υγεία του αεροσκάφους (Σχήμα 23). Για να μεταβούμε στο επόμενο επίπεδο πρέπει να καταρρίψουμε όλους τους εχθρούς που υπάρχουν στην οθόνη. Το σκορ του παίκτη υπολογίζεται κάθε φορά που συμβαίνει κάτι. Συγκεκριμένα ο παίκτης όταν

χτυπήσει έναν εχθρό με το βλήμα του κερδίζει 100 πόντους, όταν χτυπήσει έναν εχθρό με το λέιζερ κερδίζει 20 πόντους και όταν παίρνει ένα κιβώτιο υγείας κερδίζει 100 πόντους. Ωστόσο μπορεί να χάσει 100 πόντους αν συγκρουστεί με τον εχθρό. Ακόμα ο παίκτης ξεκινάει το παιχνίδι με 500 πόντους υγείας και όταν αυτοί μηδενιστούν το παιχνίδι σταματάει και ο παίκτης χάνει. Έτσι λοιπόν όταν το αεροσκάφος μας συγκρουστεί με ένα εχθρό ή όταν μας χτυπήσει κάποιο εχθρικό βλήμα χάνουμε 50 πόντους υγείας.

Ο παίκτης κατά τη διάρκεια του παιχνιδιού μπορεί να κερδίζει μπόνους κιβώτια τα οποία θα τον ενισχύσουν στη μάχη. Τα κιβώτια είναι: το κιβώτιο λέιζερ όπλου το οποίο εμφανίζεται κάθε τρία επίπεδα και δίνει τη δυνατότητα στον παίκτη να εκτοξεύσει μια οριζόντια ακτίνα λέιζερ και να καταστρέψει όλους τους εχθρούς, το κιβώτιο πολυβόλου όπλου το οποίο εμφανίζεται κάθε τέσσερα επίπεδα και με αυτό το αεροσκάφος εκτοξεύει τρία βλήματα αντί για ένα για πέντε δευτερόλεπτα και το κιβώτιο υγείας το όποιο εμφανίζεται κάθε πέντε επίπεδα και προσθέτει 100 πόντους στην υγεία του παίκτη.



Σχήμα 22 "Το κύριο μενού του Space Invaders"

Σχήμα 23 "Ένα στιγμιότυπο μάχης στο Space Invaders"

Πως όμως υλοποιείται αυτό το παιχνίδι με Java; Πρώτα από όλα φτιάχνουμε ένα κατασκευαστή ο οποίος είναι υπεύθυνος για την δημιουργία και

την εμφάνιση του JFrame μας. Μέσα σε αυτόν ορίζουμε το μέγεθος του, το φόντο που θα έχει το πλαίσιο και άλλα. Ένα άλλο σημαντικό που γίνεται στον κατασκευαστή είναι η προσθήκη του ακροατή του πληκτρολογίου και του ακροατή κίνησης ποντικιού στο πλαίσιο μας. Με τη βοήθεια των ακροατών προγραμματίζουμε το χειρισμό του παιχνιδιού. Τέλος μέσα στον κατασκευαστή καλούμε τη μέθοδο initGame() για να δώσουμε τις αρχικές τιμές σε όλες τις μεταβλητές του προγράμματος και τη μέθοδο menu() για να εμφανίσουμε το περιεχόμενο του αρχικού μενού.

Όπως είπαμε όταν ξεκινάμε το παιχνίδι εμφανίζεται ένα παράθυρο με το αρχικό μενού του παιχνιδιού. Πατώντας το πλήκτρο ENTER ενεργοποιείται η μέθοδος keyPressed() η οποία ανιχνεύει το γεγονός και είναι προγραμματισμένη να ξεκινάει το παιχνίδι. Συγκεκριμένα αυτό που κάνει είναι να αφαιρεί της ετικέτες μενού και να προσθέτει τις ετικέτες των σκορ, να καλεί την populateEnemies() η οποία εμφανίζει τους εχθρούς σε τυχαίο σημείο στο πάνω μέρος του πλαισίου και να δημιουργεί και να ξεκινά το νήμα του παιχνιδιού.

Η ύπαρξη του νήματος στο παιχνίδι απαιτείται για να μπορέσουμε να δημιουργήσουμε κίνηση στα αντικείμενα και να υπολογίζουμε τις νέες τιμές των μεταβλητών κάθε φορά. Το νήμα βρίσκεται μέσα στην εσωτερική κλάση Play η οποία κληρονομεί το Thread. Μέσα στη μέθοδο run() τοποθετούμε ένα βρόχο while και το ορίζουμε να σταματάει όταν το Health του παίκτη είναι ίσο ή μικρότερο του μηδέν. Κατά την εκτέλεση του βρόχου γίνονται πολύ έλεγχοι και ενέργειες που καθορίζουν την πορεία του παιχνιδιού. Κάποιες από αυτά τα πράγματα που κάνει είναι: να κινεί του εχθρούς, να καθορίζει τυχαία ποιοι εχθροί θα επιτεθούν, να εξαφανίζει τους εχθρούς αν χτυπήθηκαν από τον παίκτη, να καλεί τη μέθοδο LevelUp(), η οποία αυξάνει τους εχθρούς με γεωμετρική πρόοδο, όταν έχουν καταρριφθεί όλοι οι εχθροί και να υπολογίζει και να εμφανίζει στις ετικέτες το επίπεδο, τη βαθμολογία και την υγεία του παίκτη.

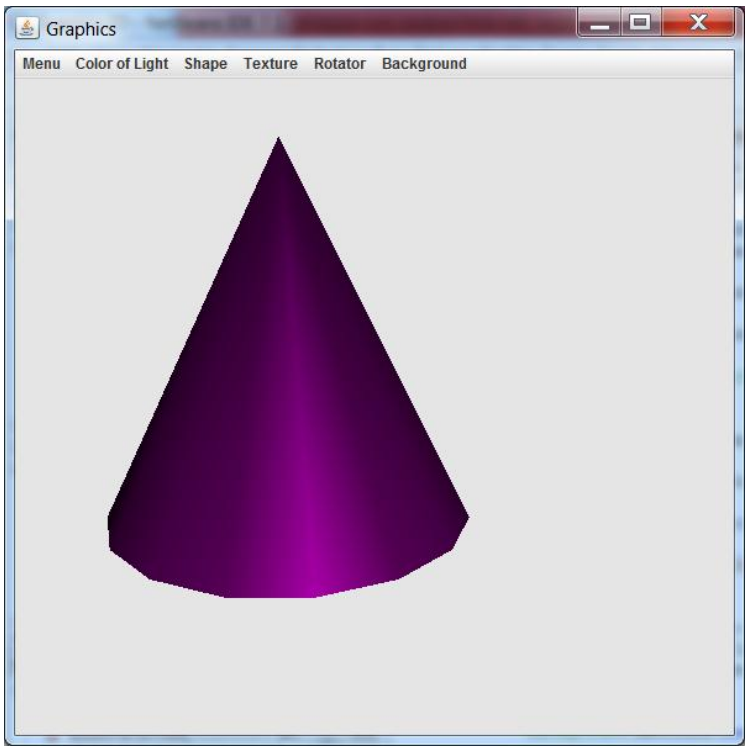
Ένα άλλο σημαντικό στοιχείο που έχει το Space Invaders είναι οι η χρήση του TimerTask για να μπορέσουμε να ελέγξουμε τον αριθμό των ρουκετών που μπορεί να ρίξει ο χρήστης όταν πατάει το πλήκτρο SPACE και να ορίζουμε το χρονική διάρκεια που θα μπορεί ο παίκτης να ρίχνει με το πολυβόλο. Έτσι λοιπόν δημιουργούμε δύο διαφορετικά TimerTasks για το καθένα στο πρόγραμμά μας, τα

οποία κάθε φορά που καλούνται αλλάζουν την τιμή μιας boolean μεταβλητής. Το πρώτο TimerTask, που καθορίζει τη συχνότητα εκπυρσοκρότησης ρουκετών, το ορίζουμε να καλείται μετά από 250 χιλιοστά του δευτερολέπτου και το δεύτερο TimerTask, που είναι ο επιτρεπόμενος χρόνος κατοχής του πολυβόλου, το ορίζουμε να καλείται μετά από πέντε δευτερόλεπτα.

Τέλος ο παίκτης μπορεί να επανεκκινήσει το παιχνίδι όταν χάσει πατώντας το κουμπί R. Όταν πατάμε το R το πρόγραμμα καλεί τη μέθοδο GameRestart() η οποία καθαρίζει τον Container από όλες τις ετικέτες, τοποθετεί όλα τα αρχικά χαρακτηριστικά και δημιουργεί ένα νέο νήμα.

ΤΟ ΠΡΟΓΡΑΜΜΑ GRAPHICS3D

Το Graphics3D είναι ένα πρόγραμμα γραφικών με το οποίο ο χρήστης μπορεί να χειριστεί διάφορα γεωμετρικά σχήματα στο χώρο, καθώς και να αλλάξει την εμφάνιση του γεωμετρικού σχήματος και του φόντου. Με αυτό τον τρόπο μπορεί να κατανοήσει τη λειτουργία του SimpleUniverse και πως κατανέμονται τα αντικείμενα μέσα σε ένα γράφο σκηνής. Επίσης δίνεται η δυνατότητα ο μετασχηματισμός ενός σχήματος όπως διαβάθμιση, επανατοποθέτηση και περιστροφή.



Σχήμα 24 "Το πρόγραμμα Graphics3D"

Πως όμως υλοποιήθηκε αυτό το πρόγραμμα; Αυτό που κάνουμε πρώτα είναι μέσα στη main να δημιουργήσουμε ένα JFrame πλαίσιο και του προσθέσουμε μία μενού μπάρα καλώντας τη setJMenuBar. Με τη κλήση της μεθόδου createMenuBar δημιουργούμε ολόκληρη τη μενού μπάρα με όλα τα JMenuItem της και ταυτόχρονα προσθέτουμε έναν ActionPerformed ακροατή σε κάθε JMenuItem. Επόμενο βήμα είναι να δημιουργήσουμε ένα αντικείμενο graphics3d καλώντας τον κατασκευαστή Graphics3D(), ο οποίος ουσιαστικά δημιουργεί το SimpleUniverse στο πλαίσιο, και να τον προσθέσουμε το αντικείμενο αυτό στο Container.

Ο κατασκευαστής Graphics3D() είναι υπεύθυνος για την απεικόνιση του SimpleUniverse στο πλαίσιο. Αρχικά δημιουργούμε ένα αντικείμενο Canvas3D με δομή γραφικών SimpleUniverse, τον τοποθετούμε στο κέντρο του πλαισίου και προσθέτουμε ένα ακροατή πληκτρολογίου για να μπορούμε να χειριζόμαστε τα αντικείμενα. Έπειτα κατασκευάζουμε ένα BranchGroup αντικείμενο στο οποίο ορίζουμε κόμβο παιδί τα αντικείμενο TransformedGroup, DirectionalLight και Background. Στο TransformedGroup δηλώνουμε κόμβο παιδί το αντικείμενο Sphere (σφαίρα), που είναι το εξ ορισμού σχήμα κατά την εκκίνηση του προγράμματος, και το αντικείμενο RotationInterpolator το οποίο περιστρέφει το σχήμα μας αν το επιλέξουμε από το μενού. Με το TransformedGroup μπορούμε να χειριστούμε όλους τους μετασχηματισμούς του σχήματος μας, με το DirectionalLight μπορούμε να ορίζουμε κατευθυντήριο φως στο γράφημα μας και με το Background διαλέγουμε το χρώμα ή την εικόνα που επιθυμούμε στο φόντο μας. Τέλος ορίζουμε ένα αντικείμενο SimpleUniverse εφαρμόζοντας του τον Canvas3D και προσθέτουμε το BranchGroup ως κόμβο BranchGraph.

Έτσι λοιπόν εκτελώντας το πρόγραμμα Graphics3D θα εμφανιστεί μία κόκκινη σφαίρα μέσα σε ένα χώρο με μαύρο φόντο. Όπως βλέπουμε στο Σχήμα 24 μέσω του JMenuBar μπορούμε να τροποποιήσουμε το σχήμα και τον χώρο όπως ακριβώς θέλουμε. Το μενού Menu περιέχει τις καρτέλες Controls και Exit. Όταν πατάμε στα Controls εμφανίζεται ένα μήνυμα διαλόγου με τα πλήκτρα χειρισμού του σχήματος. Οπότε πατώντας τα κατάλληλα κουμπιά μπορούμε να περιστρέψουμε, να αλλάξουμε το μέγεθος και να μετακινήσουμε το σχήμα μας. Πατώντας το μενού Color of Light μπορούμε να ρυθμίσουμε το χρώμα του φωτός που θα φωτίζει το αντικείμενο. Είναι σημαντικό να πούμε ότι το χρώμα κάθε

γεωμετρικού σχήματος είναι άσπρο έτσι ώστε να γίνεται ορατός ο φωτισμός πάνω του και να χρωματίζει το σχήμα ταυτόχρονα. Επιλέγοντας το μενού Shape μπορούμε να επιλέξουμε το σχήμα στο χώρα μας να είναι η σφαίρα, ο κώνος, ο κύλινδρος ή ο κύβος. Με το μενού Shape καθορίζουμε την υφή του σχήματος μας, δίνοντας το δικαίωμα να διαλέξουμε μία εικόνα από το δίσκο μας ως υφή. Στο μενού Rotator μπορούμε να ενεργοποιήσουμε η να απενεργοποιήσουμε την αυτόματη περιστροφή του σχήματος. Τέλος, το μενού Background μας δίνει τη δυνατότητα να αλλάξουμε το χρώμα του φόντου.

ΤΟ ΠΡΟΓΡΑΜΜΑ ANIMATION

Το Animation είναι ένα πρόγραμμα που παρουσιάζει μία animation ταινία διάρκειας περίπου 90 δευτερολέπτων. Όλες οι εικόνες των χαρακτήρων και του φόντου της ταινίας έχουν σχεδιαστεί με τη χρήση του προγράμματος CorelDRAW 12. Η ταινία πέρα από κινούμενες εικόνες περιλαμβάνει και ηχητικά εφέ, κάνοντας την πιο ευχάριστη και φαντασμαγορική.

Η υπόθεση της ταινίας περιγράφει ένα ανυποψίαστο πελάτη μιας τράπεζας να πέφτει θύμα ληστείας ακριβώς όταν βγαίνει από την τράπεζα. Δυστυχώς για εκείνον ένας ληστής τον παραμονεύει έξω από τη τράπεζα και με τη απειλή του όπλου του ζητάει να του δώσει ότι λεφτά εισέπραξε. Ο πελάτης εξηγεί στο ληστή ότι δεν έκανε ανάληψη χρημάτων και ότι απλά πλήρωσε τη δόση ενός δανείου που χρωστάει. Δίχως δισταγμό και υπομονή ο ληστής πυροβολεί τον πολίτη μη θέλοντας να το πιστέψει. Όμως τα πράγματα δεν εξελίσσονται όπως τα περιμένει, με αποτέλεσμα ο πελάτης να αποφεύγει τη σφαίρα και να χτυπάει το ληστή με μία εναέρια κλοτσιά (Σχήμα 25). Μετά από μερικά δευτερόλεπτα αναίσθητος, ο ληστής συνέρχεται και εξοργισμένος προσπαθεί να χτυπήσει θανάσιμα τον αθώο πολίτη. Ο πολίτης όμως γνώριζε αυτοάμυνα και με μία γροθιά κάτω από το σαγόνι ακινητοποιεί τελικά το ληστή. Για καλή του τύχη στο τέλος καταφθάνει ένας αστυνομικός ο οποίος συλλαμβάνει το ληστή.

Πως δημιουργήθηκε όμως αυτή η ταινία με Java; Όπως αναφέραμε και στο κεφάλαιο με Animation για να φτιάξουμε κίνηση εικόνων πρέπει να ορίζουμε ένα Applet το οποίο μέσω ενός νήματος θα ξανασχεδιάζεται κάθε φορά που

αλλάζουμε τις εικόνες. Στο πρόγραμμα έχει κατασκευαστεί μία μέθοδος update() για την αντιμετώπιση του τρεμοπαιξίματος της οθόνης κατά την εκτέλεση. Ουσιαστικά δημιουργεί μια offScreen στην οποία φορτώνει τις εικόνες πάνω και μόλις ολοκληρώσει το φόρτωμα τις μεταφέρει στην τρέχον οθόνη. Ακόμη ένα άλλο εργαλείο που χρησιμοποιείται είναι ο MediaTracker. Με το Mediatracker προσδιορίζουμε πότε όλες οι εικόνες έχουν φορτωθεί πλήρως στη μνήμη και εμποδίζει να σχεδιαστούν εικόνες που δεν έχουν φορτωθεί πλήρως. Η αρχικοποίηση όλων των καρέ, των ήχων και φόντου της ταινίας γίνεται στη μέθοδο init(). Επιπλέον στην init() ορίζουμε το MediaTracker, το μέγεθος του Applet. Η μέθοδος run() είναι υπεύθυνη να ξανασχεδιάζει τις εικόνες ανάλογα με τη σκηνή της ταινίας. Οι σκηνές της ταινίας έχουν τοποθετηθεί μέσα σε βρόχους while, έτσι ώστε κάθε φορά που τελειώνει μια σκηνή να βγαίνει από ένα βρόχο να μεταβαίνει στην επόμενη σκηνή while. Τέλος οι μέθοδοι repaint() και thread.sleep(χρόνος) καλούνται μέσα σε κάθε while και ανάμεσα σε δύο βρόχους για να ενημερώνουν την οθόνη σχεδιάζοντας τις νέες εικόνες και να σταματάνε για πολύ μικρό χρονικό διάστημα το νήμα αντίστοιχα.

Σχήμα 25 "Ένα καρέ της ταινίας Animation"