48. Εξηγήστε την τεχνική Antialiasing και Alpha channel που χρησιμοποιείται σε εφαρμογές επεξεργασίας εικόνας και φωτορεαλιστικής απεικόνισης αντικειμένων.

Ένα από τα σημαντικότερα προβλήματα των γραφικών είναι ότι κάθε γραμμή που είναι κυρτή ή παρουσιάζει κάποια κλίση εμφανίζει οδοντώματα, μοιάζοντας περισσότερο με μια μεγάλη σκάλα πάρα με μια ευθεία γραμμή. Αυτές οι ακανόνιστες γραμμές είναι ιδιαίτερα εμφανείς στις κινούμενες εικόνες, πράγμα που τις καθιστά ιδιαίτερα ενοχλητικές στα βιντεοπαιχνίδια. Αυτό το φαινόμενο στα 3D γραφικά είναι γνωστό ως aliasing. Το φαινόμενο εμφανίζεται διότι η εικόνα στην οθόνη είναι μόνο ένα δείγμα από pixels της αρχικής τρισδιάστατης εικόνας που η κάρτα γραφικών σας έχει υπολογίσει. Στις μεγαλύτερες αναλύσεις, καθώς χρησιμοποιούνται περισσότερα pixels, το δείγμα έρχεται πιο κοντά στην αρχική του μορφή και ως εκ τούτου η εικόνα είναι ευκρινέστερη και εμφανίζει πολύ λιγότερο aliasing.

Για να αντιμετωπιστεί το πρόβλημα χρησιμοποιείται μια τεχνική που ονομάζεται Full Scene Anti-Aliasing (FSAA) ή απλώς Anti-Aliasing (AA). Το Anti-Aliasing βοηθά στο να εμφανίζονται οι ακανόνιστες αυτές γραμμές πιο ‘’ομαλές’’, χωρίς να χρειάζεται να αυξήσετε την ανάλυση της οθόνης σας. Ουσιαστικά το AA συνδυάζει τις ακανόνιστες γραμμές με το περιβάλλον τους και μπορεί να εφαρμοστεί, είτε σε 2D ή 3D γραφικά.

Τύποι anti-aliasing

Υπάρχουν διάφορες μέθοδοι για την επίτευξη του anti-aliasing. Οι περισσότερες από αυτές βασίζονται σε δύο μεθόδους:

1. Στη μέθοδο του **sampling**: το sampling είναι μια διαδικασία όπου επιλέγεται μία μόνο τιμή από μια συνεχώς μεταβαλλόμενη σειρά τιμών. Το αποτέλεσμα του sampling είναι μία τιμή που ονομάζεται sample. Στα 3D γραφικά, ένα sample περιγράφει μια περιοχή γεωμετρίας.
2. Στη μέθοδο του **post-processing**: σε αυτή τη μέθοδο ένα αλγόριθμος τροποποιεί την εικόνα λίγο πριν αυτή προβληθεί στη οθόνη μας, αφαιρώντας, όσο aliasing μπορεί να εντοπίσει.

Η πιο κοινή μέθοδος πριν από λίγα χρόνια ήταν το **Super Sampling (SSAA)** η οποία απλά αυξάνει την ανάλυση ολόκληρης της εικόνας εντός της κάρτας γραφικών, την οποία μετά συρρικνώνει στην επιλεγμένη ανάλυση της οθόνης σας. Αυτή η μέθοδος μειώνει τις επιδόσεις σε πολύ μεγάλο βαθμό, αλλά λειτουργεί με σχεδόν οποιοδήποτε παιχνίδι, προσφέροντας τα καλύτερα αποτελέσματα.
Το **Multi Sampling (MSAA)** είναι μια νεότερη μορφή AA που βασίζεται στο super sampling anti-aliasing, αλλά αντί να αυξάνει την ανάλυση ολόκληρης της εικόνας εντοπίζει τις πλευρές που χρειάζονται anti-aliasing και επεξεργάζεται μόνο τις περιοχές αυτές. Έτσι ο αντίκτυπος στις επιδόσεις είναι πολύ μικρότερος.

49. Τι είναι το encapsulation; Δώστε ένα παράδειγμα σε γλώσσα προγραμματισμού Java.

Ο συνδυασμός δεδομένων και συναρτήσεων μέσα στα αντικείμενα, και η «προστασία» των private δεδομένων μέσω των public συναρτήσεων λέγεται ενθυλάκωση (encapsulation).

Παράδειγμα Java:

```
public class EncapTest {
   private String name;
   private String idNum;
   private int age;
```

```
   public int getAge() {
      return age;
   }

   public String getName() {
      return name;
   }

   public String getIdNum() {
      return idNum;
   }

   public void setAge( int newAge) {
      age = newAge;
   }

   public void setName(String newName) {
      name = newName;
   }

   public void setIdNum( String newId) {
      idNum = newId;
   }
}

public class RunEncap {

   public static void main(String args[]) {
      EncapTest encap = new EncapTest();
      encap.setName("James");
      encap.setAge(20);
      encap.setIdNum("12343ms");

      System.out.print("Name : " + encap.getName() + " Age : " + encap.getAge());
   }
}
```

50. Δημιουργήστε ένα πρόγραμμα σε Java το οποίο θα ζητάει από το χρήστη να εισάγει/πληκτρολογήσει μία φράση και στη συνέχεια θα εμφανίζει πόσες φορές εμφανίζεται το κάθε φωνήεν στη φράση.

```
import java.util.Scanner;
public class Test {
 public static void main(String[] args) {
      Scanner SC = new Scanner(System.in);
   System.out.print("Enter a phrase: ");
      String s = SC.next().toLowerCase();
      int vowelCount = 0;
      int count[]=new int [5];

      for (int i = 0; i<5;i++){
        count[i]=0;
      }
```

```
        for (int i = 0; i < s.length(); ++i) {
         switch(s.charAt(i)) {
              case 'a':
                      count[0]++;break;
              case 'e':
                      count[1]++;break;
              case 'i':
                      count[2]++;break;
              case 'o':
                      count[3]++;break;
              case 'u':
                      count[4]++;break;
              case ' ':
                      break;
             default:
        }
        }
        for (int i = 0; i<5;i++){
         switch(i) {
              case 0:
                      System.out.println("a->"+count[i]);break;
              case 1:
                      System.out.println("e->"+count[i]);break;
              case 2:
                      System.out.println("i->"+count[i]);break;
             case 3:
                      System.out.println("o->"+count[i]);break;
              case 4:
                      System.out.println("u->"+count[i]);break;
             default:
        }
        }
    }
}
```

51. Δημιουργήστε ένα πρόγραμμα σε Java το οποίο θα σχεδιάζει ένα κύκλο. Το χρώμα και το μέγεθος της ακτίνας του κύκλου θα καθορίζεται από τον χρήστη.

```
package circle;
import java.lang.*;
import java.util.*;
import java.util.List;
import java.io.*;
import java.awt.*;
import java.awt.event.*;
import java.awt.geom.*;
import java.lang.reflect.Field;
import java.util.logging.Level;
import java.util.logging.Logger;
public class Circle extends Frame {

 public static void main(String args[])
 {
      Circle frame = new Circle();
     frame.addWindowListener(
      new WindowAdapter()
      {
```

```
      public void windowClosing(WindowEvent we)
      {
           System.exit(0);
      }
      }
      );

     frame.setSize(400, 400);
     frame.setVisible(true);
  }
 public void paint(Graphics g) {
      Scanner SC = new Scanner(System.in);
    System.out.print("SET R: ");
      int R = SC.nextInt();
    System.out.print("\nSET COLOR: ");
      String current_color = SC.next();
      Graphics2D ga = (Graphics2D)g;

      Field field=null;
      try {
      field                                                                    =
Class.forName("java.awt.Color").getField(current_color.toLowerCase());
      } catch (ClassNotFoundException ex) {
         Logger.getLogger(Circle.class.getName()).log(Level.SEVERE,null,ex);
      } catch (NoSuchFieldException ex) {
         Logger.getLogger(Circle.class.getName()).log(Level.SEVERE,null,ex);
      } catch (SecurityException ex) {
         Logger.getLogger(Circle.class.getName()).log(Level.SEVERE,null,ex);
      }
      Color color = null;
      try {
      color = (Color)field.get(null);
      } catch (IllegalArgumentException ex) {
         Logger.getLogger(Circle.class.getName()).log(Level.SEVERE,null,ex);
      } catch (IllegalAccessException ex) {
       Logger.getLogger(Circle.class.getName()).log(Level.SEVERE,null,ex);
      }
    ga.setPaint(color);
    ga.drawOval(150,150,R,R);
 }

}
```

52. Τι είναι τα Demon Threads και τί τα Non-Demon Threads; Από που εκτελούνται και πως;

Demon είναι τα **Threads** που συνεχίζουν να ζουν ακόμα και αν ο πατέρας τους πέθανε. Αντίθετα τα **Non-Demon** πεθαίνουν όταν πεθάνει αυτός που τα δημιούργησε.
Τα Demon Threads στην Java είναι τα threads που τρέχουν στο παρασκήνιο και δημιουργούνται κυρίως από την JVM(java virtual machine) για την εκτέλεση εργασιών στο παρασκήνιο, όπως τη συλλογή σκουπιδιών(garbage collection). Τα Demon Threads τρέχουν ακόμα και μετά το τέλος του προγράμματος.Η **βασική διαφορά** μεταξύ Demon Threads και Non-Demon Threads είναι ότι η JVM δεν περιμένει κάποιο Demon Thread πριν τερματίσει η εφαρμογή, ενώ τα Non-Demon Threads τα περιμένει και δεν τερματίζει αν τα Non-Demon Threads δεν τελειώσουν την εκτέλεσή τους.

53. Δίνεται το μη ASCII αρχείο DATA.dbs όπου κάθε εγγραφή είναι αντικείμενο που έχει την ακόλουθη δομή: IBAN λογαριασμού (20 χαρακτήρες) Ανάληψη/Κατάθεση (Ό'/Υ αντίστοιχα) Ποσό συναλλαγής (δεκαδικός αριθμός) Όνομα πελάτη (40 χαρακτήρες) ΑΦΜ πελάτη (ακέραιος 8-ψήφιος).
Να γραφεί πρόγραμμα Java προσπέλασης του αρχείου DATA.dbs που να τυπώνει στην οθόνη το άθροισμα των καταθέσεων, το άθροισμα των αναλήψεων καθώς και τα στοιχεία του πελάτη με τη μεγαλύτερη ανάληψη και κατάθεση.

```
import java.io.*;
import com.linuxense.javadbf.*;
import org.xBaseJ.*;
import org.xBaseJ.fields.CharField;
import org.xBaseJ.fields.LogicalField;
import org.xBaseJ.fields.NumField;
import org.xBaseJ.fields.DoubleField;

public class JavaDBFReaderTest {

  public static void main( String args[]) {

 String IBAN[];
 String TRANSACTION[];
 Double AMOUNT[];
 String CREDENTIALS[];
 String TIN[];
 CharField iban;
 CharField transaction;
 DoubleField amount;
 CharField credentials;
 CharField tax_number;
 int numberOfFields;
 double withdrawals=0;
 double deposits=0;
 double max_withdrawal;
 double max_deposit;
 int max_withdrawal_customer;
 int max_deposit_customer;

 try{
        InputStream inputStream=null;
        try {
               inputStream  = new FileInputStream("C:/DATA.dbs");
               } catch (IOException e) {
                              e.printStackTrace();
               } finally {
                      try {
                             if (inputStream != null)
                                    inputStream.close();
                             } catch (IOException ex) {
                                    ex.printStackTrace();
                             }
               }
        DBFReader dataDB = new DBFReader(inputStream);
        numberOfFields = dataDB.getFieldCount();
        IBAN=new String[numberOfFields];
        TRANSACTION=new String[numberOfFields];
        AMOUNT=new Double[numberOfFields];
        CREDENTIALS=new String[numberOfFields];
        TIN=new String[numberOfFields];
```

```
        for(int i=0; i<numberOfFields; i++) {
               dataDB.read();
               iban  = (CharField) dataDB.getField("IBAN");
               IBAN[i]=iban.getValue();
               transaction  = (CharField) dataDB.getField("TRANSACTION");
                TRANSACTION[i]=transaction.getValue();
               amount = (CharField) dataDB.getField("AMOUNT");
               AMOUNT[i]=amount.getValue();
               credentials  = (CharField) dataDB.getField("CREDENTIALS");
                CREDENTIALS[i]=credentials.getValue();
               tax_number = (CharField) dataDB.getField("TAX_IDENTIFICATION_NUMBER");
               TIN[i]=tax_number.getValue();
        }

        max_withdrawal=AMOUNT[0];
        max_deposit=AMOUNT[0];
        for(int i=0; i<numberOfFields; i++) {
               if(TRANSACTION[i].equals('O')){
                   withdrawals=withdrawals+AMOUNT[i];
                   if(AMOUNT[i]>max_withdrawal){
                      max_withdrawal=AMOUNT[i];
                      max_withdrawal_customer=i;
                      }
               }
               if(TRANSACTION[i].equals('Y')){
                      deposits=deposits+AMOUNT[i];
                   if(AMOUNT[i]>max_deposit){
                      max_deposit=AMOUNT[i];
                      max_deposit_customer=i;
                      }
               }
        }

        System.out.println("The total ammount of withdrawals is: "+withdrawals);
        System.out.println("The total ammount of deposits is: "+deposits);
        System.out.println("The    customer    with    max    withdrawal    amount    is
"+CREDENTIALS[max_withdrawal_customer]+    "    with    TAX_IDENTIFICATION_NUMBER    "    +
TIN[max_withdrawal_customer]+" and IBAN " + IBAN[max_withdrawal_customer]);
        System.out.println("The    customer    with    max    deposit    amount    is
"+CREDENTIALS[max_deposit_customer]+    "    with    TAX_IDENTIFICATION_NUMBER    "    +
TIN[max_deposit_customer]+" and IBAN " + IBAN[max_deposit_customer]);
        inputStream.close();
  }
  catch( DBFException e) {

        System.out.println( e.getMessage());
  }
  catch( IOException e) {

        System.out.println( e.getMessage());
  }
   }
}
```

54. Περιγράψτε τη βασική δομή μιας εφαρμογής φτιαγμένης σε OpenGL.

Η βασική δομή μιας εφαρμογής σε OpenGL αποτελείται απο δύο κυρίως κομμάτια. Το πρώτο κομμάτι είναι η αρχικοποίηση όπου καθορίζονται οι παράμετροι και οι βασικές μεταβλητές απαραίτητες για την δημιουργία της σκηνής. Οι παράμετροι αυτοί σχετίζονται με το μέγεθος της οθόνης, τα χρώμματα, το είδος της σκηνής(τρείς ή δύο διαστάσεις) κτλ. Σε δεύτερη φάση μια OpenGL εφαρμογή δημιουργεί την σκηνή συνεχώς και αλληλεπιδρά με τον χρήστη. Εδώ δηλαδή, έχουμε συναρτήσεις όπου δημιουργούν κάθε ένα Pixel στην οθόνη, αλληλεπιδρούν με το πληκτρολόγιο και το ποντίκι και είναι υπεύθυνες για το resize του παραθύρου

55. Να αναφέρετε και να περιγράψετε σύντομα τα callback functions της OpenGL.

1 glutDisplayFunc
2 glutOverlayDisplayFunc
3 glutReshapeFunc
4 glutKeyboardFunc
5 glutMouseFunc
6 glutMotionFunc, glutPassiveMotionFunc
7 glutVisibilityFunc
8 glutEntryFunc
9 glutSpecialFunc
10 glutSpaceballMotionFunc
11 glutSpaceballRotateFunc
12 glutSpaceballButtonFunc
13 glutButtonBoxFunc
14 glutDialsFunc
15 glutTabletMotionFunc
16 glutTabletButtonFunc
17 glutMenuStatusFunc
18 glutIdleFunc
19 glutTimerFunc

56. Ποιες εντολές χρησιμοποιούμε για τις εξής λειτουργίες της OpenGL: α. translate β. rotate γ. scale ή stretch;

Η συνάρτηση glTranslate πολλαπλασιάζει τον τρέχων πίνακα με έναν translation πίνακα. Η συνάρτηση αυτή παράγει ένα translation πίνακα βάσει των x y z. Ο τρέχων πίνακας πολλαπλασιάζεται με αυτόν το translation πίνακα και το γινόμενο αντικαθιστά τον τρέχοντα πίνακα. Έχει τις συναρτήσεις C, void glTranslated(GLdouble x,GLdouble y,GLdouble z) και void glTranslatef(GLfloat x,GLfloat y,GLfloat z),όπου οι παράμετροι x, y, z καθορίζουν τις x, y,και z συντεταγμένες ενός translation vector.

Η συνάρτηση glRotate πολλαπλασιάζει τον τρέχων πίνακα με έναν rotation πίνακα. Η συνάρτηση αυτή παράγει μια επριστροφή της γωνίας των αξόνων περί το διάνυσμα x,y,z. Ο τρέχων πίνακας πολλαπλασιάζεται με αυτόν το rotation πίνακα και το γινόμενο αντικαθιστά