Crit C: (Word count: 809)

Technique: File Access

Success Criteria: The program can view 3D wireframes from a file.
-The Client can access files from a folder by simply typing the name of the file within the program.
-inspiration: w3 schools

Justification: File systems are useful when attempting to store data over multiple instances of code, and decompiling that data is required.
Explanation: The file is read from a preset folder on the hard drive of the computer, and is checked before being read to avoid errors.

```
public static boolean FileCallCheck(String name){
    //just checks that the file exists and outputs some basic stuff
    //about it, taken partially from W3schools as I was learning about
    //file systems.
File myObj = new File("C:\\Program Files (x86)\\AshPgms\\Wireframes\\"+name);
if (myObj.exists()) {
  return true;
} else {
  System.out.println("The file does not exist.");
  return false;
}
}

private static String FileReadToString (String name) throws Exception {
    //reads the file specified and converts it to a long string
     String data = "";
data = new String(Files.readAllBytes(Paths.get("C:\\Program Files (x86)\\AshPgms\\Wireframes\\"+name)));
return data;
}
```

Technique: File Decompiling

Success Criteria: The program can view 3D wireframes from a file. The program's files should be mostly readable by a human.
-The program will decompile the text into a readable array, and send that array to be displayed
-inspiration: brainf*ck programming language (uses a similar pointer system)

Justification: In order for the file to be readable by a human, the storage method cannot be encoded simply, so it has to have some structure, this structure must be decompressed in order to be used by the computer.
Explanation: The string acquired by the file reader is turned into a char array and parsed through, using the symbols ",", "{}", and "!" to read every number from the text file, if it's formatted correctly.

```
private static void DataInterpereter (String data){
//a program to take the string data from the file and adapt it to a
//useable miltidimensional array. Uses a pointer system inspired by
//brainf*ck to read the information letter by letter.
char[] arr=data.toCharArray();
int tempInt=0;
double[] tempArr={};
int StartPoint=0;
int Point=StartPoint+1;
char pointFind='n';
int currentEnd=0;
boolean EndFound=false;
boolean ReadNum=false;
boolean Completed=false;
boolean end=false;
while (Completed==false) {
//^This loop is to loop all of the code for multiple data sets.
while (EndFound==false&&Completed==false){
//^This loop is to find the next start of a data set.
    end=false;
    if (arr.length-1<=Point+1||arr.length-1<=Point||arr[Point]=='!'){
        /*This will detect when the pointer has reached it's end, and will
        *shut down everything and Change the PointData to a useable
        *multidimensional array. And I'm leaving the system out because I
        *think it looks like the matrix, which is cool.
        */
        System.out.println("End");
        PointData=To2d(tempArr,7);
        System.out.println(Arrays.deepToString(PointData));
        Completed=true;
    }
    if(arr[Point]=='}'&&Completed==false){
    //flips the switch to read the data in the points
        EndFound=true;
    }
    if(Completed==false){
    //progresses the pointer
        Point=Point+1;
```

(there is far more code, I just can't fit it all on here)

<u>Technique: Perspective Rendering</u>

Success Criteria: The program can view 3D wireframes from a file. The program can output instructions on where to draw lines in the game engine.

-The program has a way to calculate the position of the point on a grid which is public meaning that it can be output onto the screen of a game engine with the use of the function within the code of the game.

-inspiration: Elite for the commodore 64

Justification: In order for 3D to be achieved, perspective needs to be calculated, the way I've done this is with a system telling the engine where each point is. This system is public, so it can be implemented into my client's game should he want it.

Explanation: The code used here calculates perspective on 1 axis, it takes the x and y coordinates of the 3D model and translates them into the x coordinates of the screen, there is an identical function which takes the x and z coordinates of the 3D model and translates them to the y coordinate of the screen, and together they are able to output the x and y coordinates of a 3d point from the perspective of the camera.

```
public static double LineEqX(double X, double Y){
 /*A function which is essentially raycasting in 2D space, grab the X
  * and Y coordinates of the point you want to calculate, and have
  * them act as a representation of the slice of 3d space you want,
  * then draw a line to athe camera and find the point that it
  * intersects with a line, which will then output to a x coordinate
  * on the screen.
  */
  //getting the relative X points to the camera
  double Xc=X-camX;
  double Yc=Y-camY;
  /*a multiplication to make the point on the same grid as the fov
   *camera, based on the distance at 90, which is 50.
   *if the number is greater (or less than based upon the camera) then
   *it outputs an error, 1234567, which is interpereted as nothing.
   */
  Xc=Xc*50;
  Yc=Yc*50;
  double tempCamX=camX*50;
  double tempCamY=camY*50;
  if (camFace==4 && tempCamY<Yc){
        return 1234567;
    } else if (camFace==2 && tempCamX<Xc){
        return 1234567;
    }   else if (camFace==1 && tempCamX>Xc){
        return 1234567;
    } else if (camFace==3 && tempCamY>Yc){
        return 1234567;
    }
  //then we find the slope (rise over run), however, if it is facing
  // the y coordinate, they are swapped. Same for Y intercept.
  double Rise=0;
  double Run=0;
  double Inter=0;
  if (camFace==4){
        Run=Yc;
        Rise=Xc+fovDistanceX;
        Inter=-fovDistanceX;
    } else if (camFace==2){
        Run=Yc;
        Rise=Xc+fovDistanceX;
        Inter=-fovDistanceX;
    }   else if (camFace==1){
        Run=Xc;
        Rise=Yc-fovDistanceX;
        Inter=fovDistanceX;
    } else if (camFace==3){
        Run=Yc;
        Rise=Xc-fovDistanceX;
        Inter=fovDistanceX;
    }
    //then we find the x intercept with the equation x=-c/m, and shift
```

(there is far more code, I just can't fit it all on here)

Technique: Wireframe Drawing

Success Criteria: The program can view 3D wireframes from a file.

-The program draws the wireframe on screen by using the perspective points, and connecting them each to their specified connections, using their points too.

-inspiration: Elite for the commodore 64

Justification: In order to see and evaluate the model, it's important we draw it, which uses the perspective points, and connects the dots, with exceptions some types of line, which is specified by the code.

Explanation: Each point has 4 indexes behind it, 3 for connections, and 1 for the type of line, the code simply looks at those indexes, and uses the perspective to connect the dots.

```
        StdDraw.setPenColor(0,235,255);
    } else if (properties==2) {
        StdDraw.setPenColor(255,151,0);
    } else if (properties==3) {
        StdDraw.setPenColor(0,255,0);
    } else if (properties==4) {
        StdDraw.setPenColor(255,135,242);
    } else if(i==selectedPoint) {
        StdDraw.setPenColor(255,0,255);
    } else {
        StdDraw.setPenColor(255,255,255);
    }
    //X and Y being mapped to their point on the screen.
    double X= WireOut.getPersPointX(i);
    double Y= WireOut.getPersPointY(i);
    if(i==selectedPoint){
        StdDraw.circle(X,Y,5);
    }
    if (properties==10) {
    //if the point has a colour of 10, it is a point
    //and is rendered as such.
        StdDraw.setPenRadius(0.01);
        StdDraw.point(Math.abs(X),Math.abs(Y));
        StdDraw.setPenRadius(0.005);
    } else {
        double[] Connections=WireOut.getPointConnections(i);
        for(int j=0;j<3;j++){
            //checks the point for connections, if there is one, it then
            //locates that point in the array, and maps it to the screen,
            //then drawing a line from it to regular X.
            int P=(int)Connections[j];
            if(P!=0){
                double Xp= WireOut.getPersPointX(P);
                double Yp= WireOut.getPersPointY(P);
                if(X==1234567||Y==1234567||X==1234567||Yp==1234567){
                    //error message
                    System.out.println("a line is unable to be rendered in line "+i+" to "+P+".");
                } else {
                StdDraw.line(Math.abs(X),Math.abs(Y),Math.abs(Xp),Math.abs(Yp));
        }
    }}}}
```

(a little bit was cut off the top)

<u>Technique: Planar Rendering</u>

Success Criteria: The program can view 3D wireframes from a file. The program has the tools to edit 3D wireframes, by creating points, moving them, and drawing lines in between them.

-The program has an editing mode, which cannot edit files, but does assist the user in multiple ways while editing their files directly as they are .txt files, this is easy. The client agrees on this part.

-inspiration: picoCAD

Justification: Seeing the model from a fixed, non-perspective view can help when developing the model, seeing the points line up, as well as how they connect.

Explanation: The code here is a simplified version of displaying the 3D wireframe, where the X, Y, and Z coordinates are taken directly instead of going through perspective changes.

```
private static void DrawPlane(int xOffSet, int yOffSet, String plane){
    //will automatically scale down to 1 quarter of the screen
        int num=WireOut.getPointAmount();
        for(int i=1;i<num;i++){
            double properties=WireOut.getLineProperties(i);
        if (properties==1){
        //reads the last number in a given point, and sets the colour
        //based on predetermined variables.
            StdDraw.setPenColor(0,235,255);
        } else if (properties==2) {
            StdDraw.setPenColor(255,151,0);
        } else if (properties==3) {
            StdDraw.setPenColor(0,255,0);
        } else if (properties==4) {
            StdDraw.setPenColor(255,135,242);
        } else if(i==selectedPoint) {
            StdDraw.setPenColor(255,0,255);
        } else {
            StdDraw.setPenColor(255,255,255);
        }
        if(plane=="Top"){
            double X=((WireOut.getPointX(i)-WireOut.getCenterX())*EditScale)+xOffSet;
            double Y=((WireOut.getPointY(i)-WireOut.getCenterY())*EditScale)+yOffSet;
            if(i==selectedPoint){
                StdDraw.circle(X,Y,5);
            }
            double[] Connections=WireOut.getPointConnections(i);
            for(int j=0;j<3;j++){
                int P=(int)Connections[j];
                if(P!=0){
                    double Xp=((WireOut.getPointX(P)-WireOut.getCenterX())*EditScale)+xOffSet;
                    double Yp=((WireOut.getPointY(P)-WireOut.getCenterY())*EditScale)+yOffSet;
                    //System.out.println("Drawing line from: "+X+", "+Y+" to "+Xp+", "+Yp);
                    StdDraw.line(Math.abs(X),Math.abs(Y),Math.abs(Xp),Math.abs(Yp));
                    StdDraw.setPenColor(255,255,255);
                    StdDraw.textRight(xOffSet*2,yOffSet*2,plane);
            }
        }
        }
        if(plane=="Front"){
```

(yet again, more code, but no space)

Technique: Controls for selection

Success Criteria: The program can view 3D wireframes from a file. The program has the tools to edit 3D wireframes, by creating points, moving them, and drawing lines in between them.
-The program can detect keypresses, this is used to select different points on the 3D Model, and see their connections, as well as their coordinates on the bottom of the screen
-inspiration: picoCAD

Justification: Being able to select individual points is almost necessary when making a model. Selecting points that are off helps the process, when hunting for imperfections.
Explanation: The selectedPoint is used throughout other functions to highlight whichever point is selected, and StdDraw, the extension that I'm using features a key reader, so, I use that to read what key is being pressed, and control the editor.

```
        //right
        if (StdDraw.isKeyPressed(39)==true){
            rightPressed=true;
        } else {
            rightPressed=false;
        }
        //Z
        if (StdDraw.isKeyPressed(90)==true){
            ZPressed=true;
        } else {
            ZPressed=false;
        }
        //X
        if (StdDraw.isKeyPressed(88)==true){
            XPressed=true;
        } else {
            XPressed=false;
        }
        if(cooldown==false){
            if (rightPressed==true){
                cooldown=true;
                WireOut.setSelectedPoint(WireOut.getSelectedPoint()+1);
                selectedPoint=WireOut.getSelectedPoint();
            }
            if (ZPressed==true){
                cooldown=true;
                WireOut.MoveAll();
            }
            if (leftPressed==true){
                cooldown=true;
                WireOut.setSelectedPoint(WireOut.getSelectedPoint()-1);
                selectedPoint=WireOut.getSelectedPoint();
            }
        }
        if(cooldown==true&&leftPressed==false&&rightPressed==false){
            cooldown=false;
        }
    }
```

(some of the controls got cut off, but the important part is there)