CPSC 314
Midterm Solutions

March 2008

1. a) (2,4,6) (2 pts).

   b) True. (2 pts).

   c) Solution given by the following matrix (4 pts):

$$\begin{pmatrix} 1/2 & 0 & 0 & -2 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

   1 pt deduction per incorrect entry or
   2 pt deduction for wrong translation/scale triple.

   d) False. This approach won't work for concave polygons. (2 pts).

   e) False. Matrix multiplication is not commutative in the general case. (2 pts).

2. a) A (2, -1.5) B (-1, -4) (5 pts).

   b) The solution is (8 pts):

$$\begin{pmatrix} 0 & 1 & 0.5 \\ 1 & 2 & -3 \\ 0 & 0 & 1 \end{pmatrix}$$

   Right-multiplying this matrix by (2, -1.5, 1) gives the solution (-1,-4,1).
   Note that the goal is not to reconstruct a series of basic transformations.

   1 pt deduction if there was no verification step
   2 pt deduction for incorrect columns in the matrix
   3 pts awarded for trying to express vectors I_A, J_A in terms of I_B, J_B.

   c) The solution is (8 pts):

```
glTranslate( 4, -1, 0 );
glScale( 1, 1, 2 );
glRotate( 90, 0, 0, 1 );
```

   2 pt deduction for wrong order of rotate and translate.
   1 pt deduction for scale by zero.

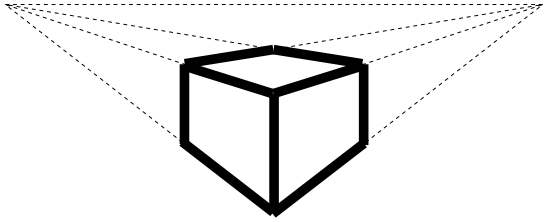
3. a) A E F G. (7 pts).

   3 pts awarded if the multiplications were correct but the graph walk was backwards.

   b) G^{-1} F^{-1} E^{-1} B C D. (7 pts).

   3 pts awarded if the multiplications were correct but the graph walk was backwards.

4. a) Figure should look as follows (2 pts):



   Drawing should depict a cube with two vanishing points or one obvious axis of which edges are parallel.

   Explanation: align image plane with aforementioned parallel axis (2 pts).

   b) Observe that...

   x' = x – (a * y) / b (3 pts awarded)
   y' = 0 (1 pt awarded)

   The required matrix is... (2 pts)

$$\begin{pmatrix} 1 & -a/b & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

   2 pts awarded for good system of equations but no final answer.

5. a) $(x-C_x)^2+(y-C_y)^2=R^2$ (3 pts)

   1 pt deduction if C_x/C_y are missing.

   b) $x=C_x+r\cos(t),\ y=C_y+r\cos(t)$ (3 pts)

   1 pt deduction if C_x/C_y are missing.

c) The best approach to this question is to maintain integers representing the row and column and to increment either one (just row) or both (row and column) based on a decision variable, e.g... (12 pts)

Floating point form...

```
x = -R
y = 0
d = f( x+0.5, y-1 )
SetPixel(Cx + x, Cy + y)
while(( y > -sqrt(2)*R/2 )&&(x < -sqrt(2)*R/2))
    if (d < 0){ //in circle
        d = f(x + 0.5, y - 2) // update the decision variable
    }else{ //outside
        d = f(x + 1.5, y - 2) // update the decision variable
        x++
    }
    y--
    SetPixel(Cx + x,Cy + y) // draw the new point
}
```

Optimized integer form...

```
x = -R
y = 0
d = 1 - R  //solve f(x+0.5,y-1) above
SetPixel(Cx + x, Cy + y)
while(( y > -sqrt(2)*R/2 )&&(x < -sqrt(2)*R/2))
    if (d < 0){ //in circle
        d = -2*y-3 // update the decision variable
    }else{ //outside
        d = 2 * (x - y) -1 // update the decision variable
        x++
    }
    y--
    SetPixel(Cx + x,Cy + y) // draw the new point
}
```

A more complete explanation of how this is derived can be found at:
http://www.ugrad.cs.ubc.ca/~cs314/Vsep2005/quiz2/q2_sol.pdf

Other correct approaches are possible. Part marks were awarded based on the use of a decision variable, correct initalization, looping, feasibility/efficiency of the solution, level of detail of the solution, and diagrams.

d) The general idea is simply to loop over a bounding box surrounding the circle and evaluate the circle function at each point to see if it is inside or not:

```
y = Cy - R
x = Cx - R

while(y < Cy + R)
    while(x < Cx + R) {
        if ((x - Cx)^2 + (y - Cy)^2 < R^2)
            SetPixel(x, y)
```

```
        x++
    }
    y++
}
```

Points were awarded for correct looping, initialization, and function evaluation. Alternative algorithms were given part marks based on how reasonable and how well-specified they are. Points awarded roughly break down as follows:

Solution equivalent to what is shown above .... 10 pts
Well-specified, feasible solution .... 6 pts or more
Poorly specified or somewhat infeasible solution .... 4 pts or less

e) In contrast with (d), the idea with (e) is to fill in the circle row-by-row and avoid evaluating the circle function at each step. Schemes involving booleans keeping track of whether or not the current point is inside/outside the circle are not necessary as with general polygons (where edges can cross). The best solution is to solve for bounding X values at each row and fill in the pixels between them, e.g.

```
for( int row = Cy - R; row <= Cy + R; row++ ) {
   int minX/maxX = Cx +/- sqrt( r^2 - y^2 )  //solve for x

   for( int col = minX; col <= maxX; col++ ) {
      SetPixel( x, y )
   }
}
```

Again, there were many possible solutions. Points were awarded in a way similar to (d). Presenting exactly the same algorithm for both (d) and (e) resulted in deductions.