CMSC425 Spring 2019
Homework 1: Geometric exercises (Corrected)
Assigned Monday, Feb . 25th
Due by midnight on ~~Sunday, March 3rd~~ Revised to Monday, March 4th
Submit PDF on Elms (no paper required)

Part a. Warm up problems
These are intended as straightforward use of basic formulas to review and debug your understandings of the concepts. You may use Octave or another vector calculator, but should show the steps you use. Notice that the problems mix ordinary text (p=(1,2)) and MS Word equation mode ($v^{\perp}$). You can use either mode, or Latex, if the answer is clear.

1. Given the points p1=(1,2) and p2=(9,7), give the point-vector from of the ray originating at p1 and going through p2. For what value of t is the point (21,14.5) on the line?

2. How far is C=(6,11) from the line through A=(2,5) and B=(4,-1)?

3. Given the two vectors a=<1,2> and b=<3,4>, give u1 and u2 in the orthogonal projection of a onto b.

4. Give the angle between the two vectors u=<-1,1,2> and <-1,0,1>.

5. Given the three points P1=(1,1,1), P2=(1,2,1), P3=(3,0,4), give the point-normal form of the plane through them using the centroid of the triangle as the point.

6. What is the value of the cross product if all three vectors are colinear (eg, parallel)?

7. For a vector v=<x,y>, the 2D perp vector $v^{\perp}$ can be defined as vperp=<-y,x>. Will this vector always be 90 degrees counterclockwise from v?

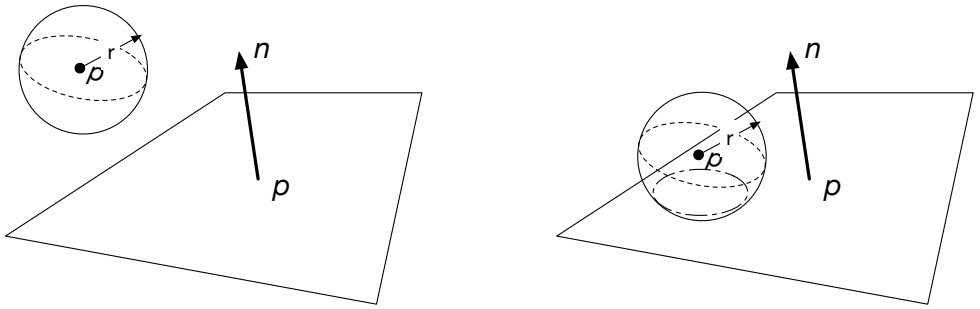
8. The perp vector $v^{\perp}$ is well defined for non-zero 2D vectors, but not for 3D vectors (why?). For a 3D vector v = <x,y,z> with all three components non-zero, can you give an effective perp vector that is orthogonal to v? What happens if one (only one) of x,y,z is zero? What happens if two (only two) of x,y,z are zero?

Part b. Applications
These are intended as applications of the formulas to game design problems.

1. You want to design a sports game with a ball, but the arena is complex and the wall may be at arbitrary angles (the arena is not just a floor with four vertical walls). Think of racquetball in an icosahedron in space. You need to implement a collider algorithm that detects when a ball (represented by a sphere) collides with a wall (represented by a plane in point-normal form.)

The normal vector points in the direction the front face of the wall. We also want to detect if we collide with the front or rear face, so when a collision happens if the center of the ball is on the front side, report "front hit", if the center is on the rear side report "back hit", and if there is no collision still report "front no hit", or "rear no hit". So there's four outcomes depending on the side of the wall. (Of course, in the space example we hope it's not outside…)

Assume the plane is given by the normalized normal vector $n = \langle n_x, n_y, n_z \rangle$ and the point $q = (q_x, q_y, q_z)$, and the sphere by the center point $p = (p_x, p_y, p)$ and radius r. On the left is a front no hit, and on the right a front hit.



a) Sketch a solution as a sequence of steps

b) Give a Unity method using Vector3 that implements solution. It should take the plane as point and normal, and the sphere as point and radius, and return properly coded the four cases.

2. Use the solution to Part 1(5) to give the key step in Midpoint displacement. In this step, given three 3D points p1, p2 and p3, you compute the vector perpendicular to the plane defined by the three points. Then, using the point-vector form of a line, compute a new point p displaced by a random number in the range (-d,d), which creates the three new triangles (p1,p,p3), (p3, p, p2) and (p2, p, p1). All you really need to do is create the point p from d and (p1,p2,p3).

The full algorithm is recursive. You continue by subdividing the three new triangles, and so on. But every recursive algorithm needs a base case. Figure out how, in computing the new point p, you might find the area of the triangle (p1, p2, p3) and use that to halt the recursion when that value is less than some given value, say min_area.