Math in action in a simple artillery game: Java Tanks

So here is your chance to actually implement some of that Algebra I, Geometry and Algebra II in a functioning game. Here is the shell for a basic artillery game: each player selects an angle for their tank and a power (or speed) for their shell, in hopes of destroying the enemy tank before they themselves are destroyed. Once the assignment is completed, the game will have:

-randomly generated terrain - some of which is destroyable and some of which is not.

-variable winds that can change dynamically.

-turned based rounds and real-time rounds, where each player can simultaneously target the enemy and shoot.

-bombers that traverse the skies, altering the landscape with their ordinance.

-howitzer rounds where the power is locked at full, and players must lob all of their shots with a drastic vertical arc.

-easter egg themes that pay tribute to some of the original artillery games like QBASIC gorillas and the ATARI 2600.

You, the student, will be responsible for completing some of the algorithms to get the game in good working order. Here are your tasks:

0) Compile and run TanksDriver.java. Note that when you fire, the shell does not impact with the ground, nor any other players. Also, the shell trajectory will not change if you pick a different angle and power. Lastly, the maximum power will not scale to the size of the battlefield. All of these tasks will fall to you to complete.

1) in *Ordinance.java*, complete the method distance:

//pre: all arguments are valid pixel coordinates, 0 <= x1,x2 <= screenWidth, 0<= y1,y2 <=screenHeight  
//post: returns the distance between the two points to see if there is a collision between players  
**private static double distance(int x1, int y1, int x2, int y2)**

This will be used for the collision detection in the methods *checkWallCollision*, *checkPlaneCollision* and *checkPlayerCollision*. The return statement in the shell is a temporary one to keep things compiling.

When completed correctly, compile and run TanksDriver.java. When the shell impacts the ground, it should now explode. If you get lucky and your shell hits the other player, they will explode.

(MATH LEVEL: ALGEBRA 1)

2) in *TanksPanel.java*, note the first line of the method *resetGame*(): this assigns the maximum power that you can select for a shell, and it varies depending on the size of the battlefield. The *maxPower* formula should make it such that the smallest battlefield (30x30) yields a *maxPower* of 50, and the largest (70x46) yields a *maxPower* of 100. The maxPower should scale appropriately to the size of the battlefield. The existing assignment of 50 to *maxPower* is temporary to keep things compiling.

When completed correctly, compile and run TanksDriver.java. Use the + and – keys to change the size of the battlefield to the minimum size and the maximum size, and then test the maximum cap placed on the power to see if it is scaled properly.

(MATH LEVEL: ALGEBRA 1)

3) In *Shell.java*, find the method *shoot(a, s, w)*:

//PRE: a is the angle of the shot (0<=a<360), s is the speed or power, w is the wind speed  
//POST: calculates the starting velocity from the time of the shot on the x and y planes  
**public void shoot ( int a, double s, double w)**

This method assigns the angle of the shell, the speed in which it is fired and the wind power, then computes the initial speed on the X-plane (speedX) and the Y-plane (speedY). There are temporary assignment statements there to keep things compiling. Write the code to correctly assign values to *speedX* and *speedY* depending on the angle and speed, as well as the wind power (which should be subtracted from *speedX*. Note that the argument *angle* is in degrees, and the Math library trig methods expect arguments to be in radians.

(MATH LEVEL: ALGEBRA 2 + TRIG)