Project 2

PROJECT DESCRIPTION

Imagine you're working on developing a game similar to Scorched Earth, a 2D tank-fighting game where tanks fire shells hoping to destroy the enemy tanks. One of the challenges of the game is that the player has to predict where the shell will land after following a parabolic trajectory. While you may not have taken physics, the equations are straightforward and you should be able to develop a solution that will accurately calculate the shell path given a starting shell speed and trajectory (angle of fire).

REQUIREMENTS

Physics Equations (from Physics for Game Developers by David Bourg):

theta = ?	// initial angle you'll ask the user to provide
speed = ?	// initial speed you'll ask the user to provide
$g = 9.8\ m/s^2$	// acceleration due to gravity
v_{ox} = speed * cos(theta)	// x component of the velocity at start
v_{oy} = speed * sin(theta)	// y component of the velocity at start
$t = v_{oy} / g$	// time until shell reaches apex
$h = v_{oy}^2 / (2g)$	// height of shell at apex
$d_x = v_{ox} * 2t$	// distance shell travels horizontally (assuming launch and target elevations are equal)

Using substitution, you should be able determine the distance the shell will rise and travel.

Your solution should do the following:

- Print a "welcome" message to the user telling them that this application will calculate the maximum height of the shell and the distance it will travel along the ground.
- Prompt the user for the initial angle in degrees (ask for the angle, read the value, parse the value and then store the value)
  - WARNING: Convert the angle from degrees to radians! If you don't, the cosine and sine methods you use later won't work properly.
- Prompt the user for the initial speed (ask for the speed, read the value, parse the value and then store the value)
- Calculate v_{ox} using the `Math Cos` method
  - `float vox = speed * (float)Math.Cos(theta);`
- Calculate v_{oy} using the `Math Sin` method
  - `float voy = speed * (float)Math.Sin(theta);`
- Calculate t
- Calculate h
- Calculate d_x
- Print an appropriate message and the value for h
- Print an appropriate message and the value for d_x

You may assume the following:

- All data from the user is of the appropriate type
    - theta – a floating point value
    - speed – a floating point value
- All data from the user is positive

HELPFUL HINTS

Write 5 or fewer lines of code, save, compile, test, repeat! Don't try to write a whole block of code at once – implement one small chunk at a time.

- Begin with the first part of the requirements, then implement one requirement at a time
- Use the `Console WriteLine` method to print intermediate values while you are working on your application – just delete the extra ones before turning in your work. That way, you can make sure you're calculating each value correctly as you go along
- Save Early, Save Often, Save Everywhere!

Finally, fully test your application! Try to think about all the ways that someone else might be able to break your code.