VGP126 Assignment #6 (due Wed Aug 29)

W. Dobson 8-27-2018

Write a program that creates several classes for combat flight simulator that has a composition class called: `fighter` that uses weapon objects of classes: `rocket, missile, and bomb`. For this first class assignment let the parameters be `public` in the class so that they can be set by the programmer directly. Later we will add more functions to to the classes.

Each class should have the following variables stored for each weapon object:

1. Integer location coordinates: `x, y, z`
2. Integer velocity vector components: `vx, vy, vz`
3. Floating point mass in kg of object: `mass`
4. Floating point thrust of the object: `thrust`
5. Floating point drag of the object: `drag`
6. Integer value for weapon effectiveness: `hitpower`
7. Boolean flag to indicate if weapon objects are released: `released`

The weapons classes should have constructors that are set to the same location as the `jet` object when instantiated within the `jet` class. The constructor for each weapon should have the option to initialize all its parameters (`x, y, z, vx, vy, vz, mass, thrust, drag, hitpower`) upon instantiation. Also they should have `thrust` set to 0 until it is released (set to true). This means that the weapons constructors should set a default value of false for `released`. Once `released` the `thrust` should be set to a stored/default value for that type of weapon. See table below for weapons default parameters:

Weapon Type	`mass`	`drag`	`thrust`	`hitpower`
`bomb`	550.0 kg	0.28	0	1000
`rocket`	78.0 kg	0.15	1500.0 N	150
`missile`	245.0 kg	0.18	5200.0 N	560

The `jet` class should have a constructor that has default values as well an an option to initialize all its parameters upon initialization. The constructor should set the location and velocity parameters to be the same for each weapon object in the class. The default `jet` parameters should be:

	`x`	`y`	`z`	`vx`	`vy`	`vz`	`mass`	`drag`	`thrust`
`jet`	1500	6800	9800	120	80	0	18000	0.48	200,000

The jet class should include the class functions:

1. A function to check the range of another jet object

The distance checking function uses a pointer to another jet class to compare locations for range for checking. This returns a 1 if the jets are within the weapon range otherwise it returns a 0.

```
int inrange((jet *enemyjet, float weaponrange){  }
```

To check the distance use the *Pythagorean* distance of the differences of the X and Y coordinates for each ship. You will need use square root function in the `<math.h>` library to take the square root of the sum of the differences squared. Note that the coordinates should be cast as `(float)` to do this calculation. See example on next page:

```
// Pythagorean distance in 2D example
int x1 = 5, y1 = 10;
int x2 = 2, y2 = 6;
float d;

// Notice that the square of int differences are cast as floats for the sqrt
d = sqrt( (float)((x1-x2)*(x1-x2)) + (float)((y1-y2)*(y1-y2)) );
// now the distance d can be compared with the float collision range.
```

2. Functions to release each weapon type. These functions should return a 1 if the object was not previously released (that is `released == false`) otherwise it returns a 0 to indicate out of weapon type.

Example function prototypes: `int firerocket() {}` or `int dropbomb() {}`

3. Write a function in the `jet` class called: `void displaystatus()` that prints out the jet's 9 parameter values to the screen in a single line with each value labeled.

4. In the `main()` program body create 2 instances of the jet with jet1 at the default values and jet2 using these parameters.

	x	y	z	vx	vy	vz	mass	drag	thrust
jet2	2700	7600	9700	-120	-80	0	18000	0.48	200,000

Notice that the velocities and locations have the 2 jets approaching each other.

Then write some code to exercise the functions in the classes to make sure the status is reported correctly and that the `released` flags are set correctly for each weapon.

Be sure to document your code with a comment header with the course number, assignment name, your name, date, and a basic description of the program. There should also be a comment header for each function with a description along with comments within the code body and particularly for the *if / else if* and *while* control statements.

Grading rubric: Documentation 20%, Code compiles without error 40%, Code functionality 40%

Submit your C++ source code to the Assignment 6 Drop Box in the Brightspace - Learning Path / Week 5 module.