Smalltalk Programming
Lesson 23

In the last lesson you added the ability for `ShooterGame` to display a functioning score. You noticed that even though the score works – it does not update when enemies are destroyed. Today, you will make changes to have the score add points when an enemy is destroyed.

1. In the shooter game, what 2 objects would be responsible for adding a score when an enemy is destroyed?

2. An easy first answer is the score object. The `Score` class keeps track of the points, so it will most definitely need to be used.

3. For the second, you might have to remember which object handles the destruction of the enemy. Do you remember which object sent a message to the enemy to remove itself from the game?

4. You may have remembered that in `Shot>>hitEnemy:`, the enemy is removed from the game by sending `delete` to the enemy instance. Look at `Shot>>hitEnemy:` to see where you removed the enemy from the game.

5. The Score class is used to handle the points, but instances of `Shot` do not have a way to communicate with the score instance that `ShooterGame` initialized. However, this is not a problem. One way to solve this is to create a method in `ShooterGame` that will allow your `Shot` instance to communicate with the score instance. Type and save `ShooterGame>>points:`.

```
points: anInteger

    score points: anInteger
```

6. The method name does not have to be `points:` and it could be named anything appropriate. Since `Score` uses `points:` to add points to the score, it would be less confusing to have `ShooterGame` use the same method name.

7. Now that `ShooterGame` has a way for the shot to notify the score instance of new points, type and save the following code change in your `Shot` class.

```
hitEnemy: enemy
   owner points: 100.
   self delete.
   enemy delete
```

8. Do you remember what happened before you needed to add “^” to “self delete” in `Shot>>checkContact`? What problem did this change fix?

9. If you remembered that a deleted morph no longer has an owner – great job! In programming, you will often need to consider situations like this. The reason “owner points: 100.” needs to be before “self delete” is because the shot instance would no longer have an owner after it is deleted. The shot instance would no longer be a part of the game.

10. Test your score by adding points to your score display. Did it work?

11. What else needs to be done for your shooter game?

12. Save and Quit your Smalltalk image.