# Smalltalk Programming
# Lesson 25

In this last planned lesson, there is one more change in order to complete the designed game. The enemies should automatically be restarted without having to manually start them yourself. Today, in this final lesson, you will make the change to make this happen in your game.

1. There is already a method to generate new enemies. What is it? How might it be able to be used to automatically generate new enemies?

2. The method that exists to generate enemies is `ShooterGame>>initializeEnemies`. You sent this message before in your explore morph to generate enemies. You also added code to use the keystroke “r” to reset the enemies using this method.

3. How might this method be used to automatically generate enemies when none exist? What have you used in your `Enemy` and `Shot` classes that Morphic provides to automatically run code at regular intervals? How might this Morphic feature be used to automatically generate enemies?

4. In Lesson 15 you learned about the Morphic animation system that uses `step` and `stepTime` to run code at regular intervals. You will only need to use `step` to automatically generate new enemies. Type the code below in `ShooterGame` and save it.

**step**

```
self enemies isEmpty
    ifTrue: [self initializeEnemies]
```

5. Look at each line of code. Can you figure out what each line of the new code is doing?

6. The `ShooterGame>>enemies` method searches submorphs of `ShooterGame` to see how many enemies exist. If the *collection* of enemies is empty (enemies = 0), new enemies are then generated using `initializeEnemies`.

7. You may have noticed that `select:` is used in `ShooterGame>>enemies` while `do:` is used in `Shot>>checkContact`. Both methods can be used. Using these two different methods helps you become familiar with a couple of *collection* methods. Remember, collections are what Smalltalk uses to collect objects together into a list or group. The message `enemies` returns a type of collection. This is why either `select:` or `do:` can be used. These two methods look at all of the objects in a collection.

8. Test your changes. Does it work? Why or why not?

9. Remember, by default, when `step` is added, existing morphs are not automatically started. A message needs to be sent to those morphs in order for them to begin stepping.

10. Run the expression "self startStepping." on `ShooterGame` in your explore morph window to enable stepping for `ShooterGame`.

11. Test your changes. Does it work?

12. If you would like a longer delay to generate new enemies, what have you learned that can be used to do this? Hint – what is the other Morphic animation method that you learned in addition to the `step` method?

13. Great job with coding an entire computer game! This can be just the beginning. What else might be nice to have for your shooter game?

14. Save and Quit your Smalltalk image.