SIT771 Object-Oriented Development

Distinction Task 6.2: Robots Dodge Changes

Focus

Make the most of this task by focusing on the following:

- Process:

  Focus on consolidating your understanding of using a responsibility-driven design approach in your code alongside learning how to code multiple classes with different responsibilities error and exception-free using core OOP concepts to perform desired outcomes.

Overview

This is the fifth of a series of tasks in which you will develop a small program. These tasks are designed to help you explore the concepts being covered and to practice your programming skills.

The material in Course 3, Week 2 will help you with this task.

In this task, you will design and implement some new features for the RobotDodge game.

Submission Details

Submit the following files to OnTrack.

- A short design report, highlighting the changes you made.
- The program's code (*Bullet.cs*, *Program.cs*, *Player.cs*, *Robot.cs*, and *RobotDodge.cs*)
- A screenshot of your program running.

You want to focus on communicating your design changes, as well as iteratively adding these new features to your program. You will need to use what you have learned from the previous iterations of this task.

Instructions

The RobotDodge game has been progressing well, now the client has seen what we have produced they want to make some new additions. You have been tasked with designing and implementing these changes.

The new features are:

- The player should have lives that are lost when they collide with a Robot.
  - Their lives should be shown on the screen. Something like small heart pictures, drawn on the edge of the screen.

    - Start with 5 lives.
    - Quit the game with the player who has no lives left.
- The Player should have a score, that increases as time passes.
    - Show the score somewhere on the screen.
    - The player should get 1 point for every second that passes. So, the longer they survive the higher their score.
- The Player should be able to shoot a bullet.
    - Clicking the mouse will shoot a bullet from the player toward the mouse.
    - The bullet should be able to *destroy* one robot, then disappear itself.
    - The bullet can be drawn as a circle. (feel free to pick any color)

Here is some guidance on how to approach this task, this outlines the expectations for the design document and the code.

1. Create a UML Class Diagram for the solution. You can start by copying the existing design, then add in the new responsibilities that you want.

   We recommend that you use Lucidchart for this. You can get a pro account with your Deakin email address.

2. Think carefully about where to allocate the different responsibilities.

3. Splashkit includes a `Timer` class that you can use to keep track of time. Here is some example code that works with a Timer:

```
Timer myTimer = new Timer("My Timer");
myTimer.Start();

SplashKit.Delay(2000);
Console.WriteLine($"{myTimer.Ticks} milliseconds have passed");
Console.WriteLine($"which is {myTimer.Ticks / 1000} seconds");
```

4. Review how the Robot targeted the Player for ideas on the bullet.

5. You can test circle collision using. `SplashKit.CirclesIntersect`

The design document needs to communicate how bullets have been added to the solution. You will needto provide the overall UML Class Diagram along with accompanying descriptive text. You can also include other sequence diagrams to aid your communication. For the report you want to focus on the following:

- The details of how the bullet works.

  This is the most complex part of the design. Focus on the following:

    - How are bullets modelled
    - How do bullets move?
    - How do bullets destroy robots?

Write a report for a knowledgeable developer. Write it as if you were describing this to the unit's teaching staff. Remember to keep in mind what you are aiming to communicate: which in this case is how you changed the design to incorporate bullets. So, avoid explaining concepts that the reader will already understand, and focus on communicating your solution.

It would be a great idea to share your design with the teaching staff **before** you start coding it. Email it to your tutor and ask for their feedback, they should get back to you quickly with some ideas. Once you have the green light, go ahead and code up your new additions!

When you are finished with the design document and the code, submit your work to OnTrack for feedback.