SIT771 Object Oriented Development

Credit Task 7.2: Different Robots

Focus

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

- Concept
  Focus on mastering inheritance and its application to create dynamic gaming environments, leveraging the power of hierarchical code organization for enhanced game development.

Overview

This is the last 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 4, Week 1 will help you with this task.

In this task you will use inheritance to create a number of different kinds of Robots in the RobotDodge game.

Submission Details

Submit the following files to OnTrack.

- The program's code (*Robot.cs* and *RobotDodge.cs*)
- A screenshot of your program running

You want to focus on the use of inheritance, and the ability for the game to easily work with different kinds of robots.

Instructions

The final changes in RobotDodge will give the game two different kinds of robots. Using inheritance we will change the Robot to an abstract class, and provide concrete Boxy and Roundy classes which change the way the robot draws.

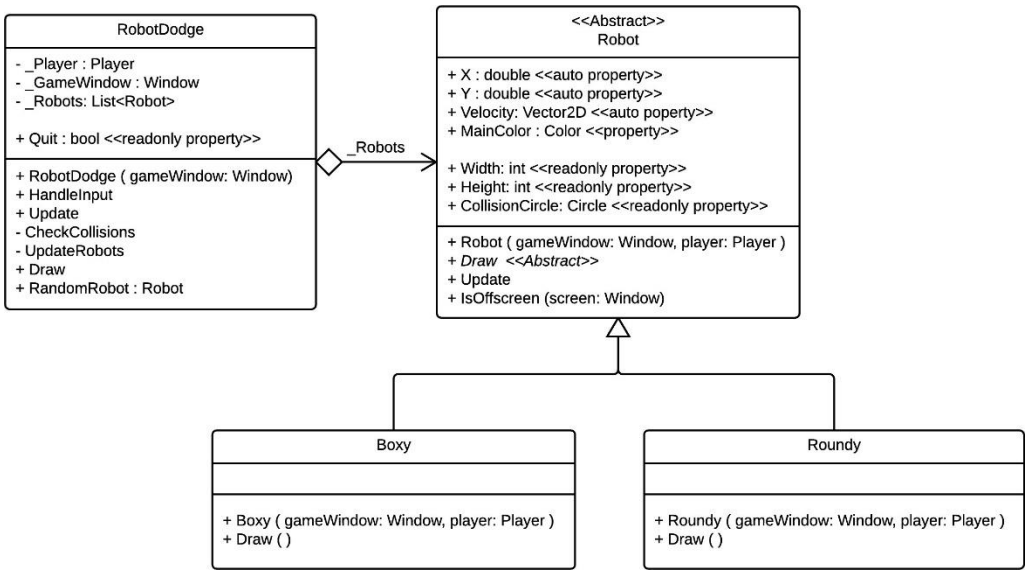

Figure: Robot Dodge iteration 4 design

1. Open your **Robot.cs** file and add a new class **Boxy**.
2. Have this new classes inherit from the **Robot** class.
3. Move the `Draw` method code from the **Robot** class into the **Boxy** class, and add a new **abstract** `Draw` method to the **Robot** class.
4. Change the **Robot** class to also be abstract.
5. Now we also need to add a **constructor** to the **Boxy** class. This will need to accept a Window and pass this on to the Robot constructor (i.e. to Boxy's **base** class constructor).
6. Finally, switch to **RobotDodge** and change the `RandomRobot` method to return a new **Boxy** object.

You should now be able to run the program and have it work as it was before.

Adding Roundy

Now we can easily add different kinds of Robots.

1. Create a **Roundy** class in the **Robot.cs** file.
2. Add an appropriate constructor.
3. Use the following code for the Draw method.

```
public override void Draw()
{
    double leftX, midX, rightX;
    double midY, eyeY, mouthY;

    leftX = X + 17;
    midX = X + 25;
    rightX = X + 33;

    midY = Y + 25;
    eyeY = Y + 20;
    mouthY = Y + 35;

    SplashKit.FillCircle(Color.White, midX, midY, 25);
    SplashKit.DrawCircle(Color.Gray, midX, midY, 25);
    SplashKit.FillCircle(MainColor, leftX, eyeY, 5);
    SplashKit.FillCircle(MainColor, rightX, eyeY, 5);
    SplashKit.FillEllipse(Color.Gray, X, eyeY, 50, 30);
    SplashKit.DrawLine(Color.Black, X, mouthY, X + 50, Y + 35);
}
```

4. Switch to **RobotDodge** and have `RandomRobot` create a Boxy 50% of the time and a Roundy 50% of the time.

5. Build and run your program.

Notice how little code we needed to change to add this new behavior. Drawing, moving, and updating the robot objects did not change! The RobotDodge game still has a list of Robots, just now that list contains Boxy and Roundy objects...

Add another robot...

Finish this iteration by adding your own special kind of robot. Adjust the program so that each Robot has a chance of being generated.

When you are finished, test that everything works, then back up your work and submit this to OnTrack for feedback.