Project Increment 1 Steps

Step 1: Add burger

After: Chapter 5

For this step, you're adding the burger to the game. You won't be able to control it yet, but at least you'll be able to see it.

1. Add code to the `Game1` `LoadContent` method to assign the `burger` field (already declared in the `Game1` class) to a newly-constructed `Burger` object. The first argument in your call to the constructor should be `Content`. The third and fourth arguments should center the burger in the center of the window horizontally and 1/8 of the window height above the bottom of the window. For now, pass `null` for the fifth argument (we'll replace this in Project Increment 5).
2. Add code to the `Burger` `Draw` method to have the burger draw itself. Note: This method is in the `Burger` class contained in the Burger.cs file.
3. Uncomment the code in the `Game1` `Draw` method that calls the `burger` `Draw` method

When you run your game, you should see a burger in the window.

Step 2: Add bear

After: Chapter 5

For this step, you're adding a teddy bear to the game. It won't be moving yet, but you'll see it in the window

1. Add code to the `Game1` `LoadContent` method to spawn a single teddy bear. Do this with the following code:

   ```
   SpawnBear();
   ```

2. Add code to the `Game1` `SpawnBear` method to generate random x and y locations for the bear using the `GameConstants` `SpawnBorderSize` constant and the `Game1` `GetRandomLocation` method. The `GameConstants` `SpawnBorderSize` constant tells us how much space to leave at all four edges of the window; you should NOT spawn a new bear in that space
3. Add code to the `Game1` `SpawnBear` method to generate a random speed using `GameConstants.MinBearSpeed`, `GameConstants.BearSpeedRange`, and the `RandomNumberGenerator` `NextFloat` method
4. Add code to the `Game1` `SpawnBear` method to generate a random angle using `Math.PI` and the `RandomNumberGenerator` `NextFloat` method (you'll need to cast `Math.PI` as a `float` to use it in your argument in the method call)
5. Add code to the `Game1` `SpawnBear` method to create a new `Vector2` object for the bear velocity using the random speed and angles you generated and the appropriate trigonometry (with more type casting here because the `Math.Cos` and `Math.Sin` methods return `double`)
6. Add code to the `Game1` `SpawnBear` method to create a new teddy bear (name the variable that holds the new object `newBear`). For now, pass `null` for the last two arguments (we'll replace those in Project Increment 5).
7. Add the new bear to the list of bears included in the game. Do this with the following code:

   ```
   bears.Add(newBear);
   ```

8. Add code to the `TeddyBear` `Draw` method to have the teddy bear draw itself. Note: This method is in the `TeddyBear` class contained in the TeddyBear.cs file.

When you run your game, you should see a burger and a teddy bear in the window. If you run it a few times, you should see the teddy bear appear in different locations in the window.

<u>Step 3: Move bear</u>

After: Chapter 5

For this step, you're making the teddy bear move in the game.

1. Add code to the `TeddyBear` `Update` method to move the teddy bear based on its velocity and the elapsed game time in milliseconds. The `Update` code just after Figure 16.17 in the book shows how you can implement this piece:

```
drawRectangle.X += (int)(velocity.X *
    gameTime.ElapsedGameTime.Milliseconds);
drawRectangle.Y += (int)(velocity.Y *
    gameTime.ElapsedGameTime.Milliseconds);
```

When you run your game, the teddy bear should be moving around. If you run it a few times, you should see the teddy bear moving in different directions.