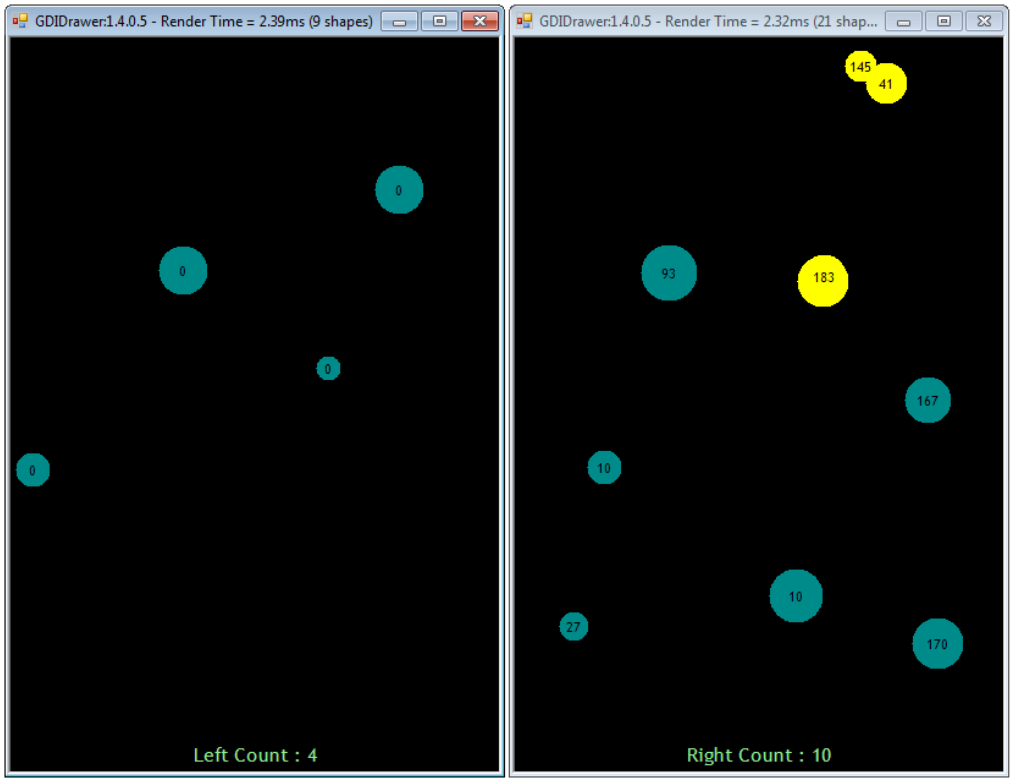
ICA #05 - CMPE2300 – My First Equals



Create another Windows Forms application that once again contains bouncing balls…

This time you will use two drawer windows to display two separate lists of your ball type.

Ball Class

Create a ball class that contains the following fields:

- `PointF` for position
- `PointF` for direction (decomposed X/Y values to add to position each tick)
- `float` for radius
- Constant `int`s for height and width of the bounding drawer window for the ball
- `int` for number of `Equals()` calls in this object that returned true
- `bool` as a highlight flag (automatic property, public set, private get)

Add a constructor that will accept a starting position, direction, and radius. Initialize all class members.

Add a `public static` helper method that can return the distance between two balls.

Add a `public Move` method that will move/bounce the ball. Adjust the position by the direction components. If the ball moves even partially out of the bounds of the drawer window, return it to edge-of-bounds and flip the sign of the appropriate direction component. Balls must always be fully visible.

Implement the `Equals` override for your ball type. `Equals` for your ball type will return true if the invoking and argument balls overlap to any extent. If equal, increase the equal count for the *invoking* object.

Add a `Render` method that will accept a `CDrawer` object reference and will render the ball to it. The ball will be rendered in yellow if highlighted, or dark cyan if not. The ball will be centered on its position, and will be drawn 2r in height and width. Use a one pixel white border on each ball. Add text for the found-to-be-equal count.

Main Form

To your main form, add two `static CDrawer` instances, 400px wide, and 600px tall, continuous update off. *Label and programmatically position them in such a way that you can tell them apart.*

Add a `static Random` object.

Add two `List`s of `Ball`, naming them somewhat consistently with the drawer windows.

In the form constructor, subscribe to the `MouseLeftClick` events in both drawer windows, adding handlers as the IDE helpers will. Use these handlers to add new balls to the lists (clicks in the left drawer window add to the left list, and clicks in the right drawer window add to the right list). New balls will have random floating point speeds and directions in the range -5 to +5 range for their X and Y components. Balls will be sized to random integral diameters in the range 20-50 pixels. Note: You are going to be modifying the lists from another thread, so for all operations that use the lists, don't forget your locks!

Add a 50ms timer to the form that will perform the ball processing. Each tick:

- Move all of the left balls.
- Move overlapping left balls to the right list: Check for overlapping balls in the left list (these are balls that are not equal by object reference, but are equal by your definition of `Equals`). You won't be able to remove the balls directly from the left list, so add them to a temporary `Ball` list. Once all the overlapping balls have been identified, remove them from the left list, and add them to the right list*.
- Clear the left drawer, draw all the left balls, and render the left drawer.
- Move all the right balls, clearing the highlight flag as you go.
- Highlight overlapping balls: Considering all balls in the right list, identify all the balls that are not equal by object reference, but are equal by your definition of `Equals`. Set the highlight flag in these balls to `true`.
- Clear the right drawer, draw all the right balls, and render the right drawer.

* Adding and removing balls from/to a list will require you to create manual code. The `List` methods '`Contains`' and '`Remove`' will use your `Equals` implementation to determine equality for their operation. You require object reference equality, so these methods may provide undesired behavior.