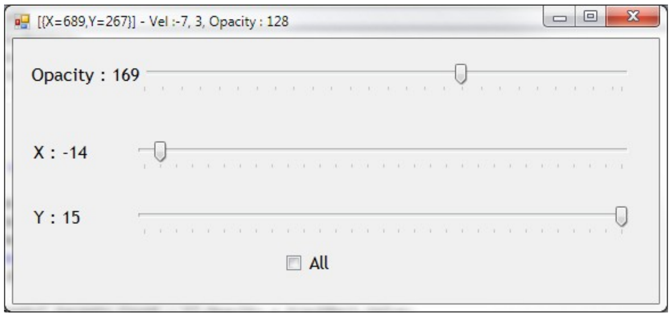
CMPE2300 – ICA02 - Bouncy Properties

In this ICA you implement a simple class representing a bouncy Ball. It will render itself, and be manipulated via properties.

Create a new Windows form project** for this ice, add the reference and supply the Drawer dll for its use

Construct a UI for your form fashioned after the following capture.:
It consists of a label + trackbar(x3) for Opacity, X and Y properties and finally a checkbox.


[(X=689,Y=267)] - Vel :-7, 3, Opacity : 128

Opacity : 169

X : -14

Y : 15

The Opacity trackbar have properties for Min/Max of 0/255
The Velocity trackbars have properties for Min/Max of -15/15

Using **Add**...new Class, create a class named Ball. Your new Ball class will contain the following instance fields :

- a static* field of Random, use an initializer
- field of **Point** representing the **<u>center</u>** of the Ball,
    - also provide a <u>manual get only</u> property called **Location**
- 2 int fields representing X and Y velocity (change per move),
    - then create a <u>manual set only</u> property for each,
        - assign X directly to the XVelocity member,
        - but restrict Y Velocity to -10 to +10
            - (ie. If Y velocity is set to -12, it is restricted to -10 max)
- field of Color representing the Ball color - no property
- a public automatic property of int representing the Ball opacity with a hidden get and public set
- a public readonly integer member representing the Ball radius

Add a **constructor** to your Ball class, accepting a Point initialize the center member using the point argument

- initialize the color with the RandColor GDIDrawer library class.
- initialize the readonly radius to 40
- initialize the X, Y velocity values to a random value between -10 and 10 <u>inclusive</u>
- initialize the opacity to 128

Now to follow the basic programming tenet of : one method = one function. We want to move all our balls **then** show all our balls… hmm, while we could do this in one method but we shall break them into 2 – one to move and one to show.

Add a public method called **MoveBall**() – this method will only move a ball to its new location. The respective velocities will be used to modify the current location. First though we must deal with "bouncing" off the walls – In order to know where the walls are we need the CDrawer object so accept this as a method argument. If adding the velocity to the current location would take it out of bounds, then change the sign of the velocity (ie. 3 to -3), you can now flip the sign of the offending velocity to the location ensuring the Ball stays in view. Perform your velocity correction <u>first</u>, now our data is valid and we can add the velocity to the Balls current location.

Add a public method called **ShowBall**() this method allows a Ball to render itself. It should return nothing and accept .. what ? Add a CenteredEllipse at the appropriate location, size and color; use FromArgb() to construct the color (ie. Color.FromArgb(opacity_value, memberColor))

Provide a ToString() override returning the string indicated in the form title - take advantage of the Point objects ToString() formatting functionality.

Now we need to bind everything together :

In your main form : add a CDrawer, a List of your Ball type, a timer (interval of 20 and enabled by default - running as soon as our application starts.)

In your form constructor allocate your CDrawer and using named parameters turn off ContinuousUpdate.

In your timer event handler we code 3 functional blocks :

- o If the user has left clicked in the drawer, add a new Ball to your List using the click Point value
- o If the user has right clicked, clear all the Balls from your List collection.
- o Clear the drawer, iterate through your list invoking MoveBall(), then ShowBall() for each Ball, <u>after</u> looping is complete remember to Render() your CDrawer to see everything.
- o If available, update the form title to the string equivalent of the <u>last</u> Ball in the list – remember there won't be one there when your application first starts.

Add Opacity Scroll handler : Update the label as shown then, Set <u>either</u> the last Ball or **all** Balls Opacity to the trackbar value depending on the checkbox

Add Velocity Scroll handlers : Update the label as shown then, Set <u>either</u> the last Ball or **all** Balls {?}Velocity to the trackbar value depending on the checkbox

Breakpoints, incremental builds and slowing the timer will all contribute to your debug “experience”

Tips :

- Code your ball class ignoring MoveBall.
- Code your main form ignoring the scroll handlers.
    - You should be able to click and add Balls to your collection and see them in the CDrawer window.
- Then implement and test MoveBall()
- Finally tie together your scroll handlers with their respective properties.