**Bouncing 2**

Modify the past bouncing program to have 10 balls. Each ball should appear on the screen in a different place.

The first click on the screen starts all 10 balls moving. When any ball hits one of the walls, it bounces as it did in the last program. Keep in mind that only the ball(s) that hit a wall should bounce; the others just keep moving until they individually hit a wall.

The second click anywhere on the screen stops all the balls. The third click closes the window and ends the program.

You can choose any original location and direction for each ball with one exception – your choices cannot be made so that all the balls move and bounce as one; as if they were all part of the same object. For example, if the balls are created in a vertical line and all of them move directly to the right at the same speed, all of them will hit the wall at the same time and bounce at the same time, and since they are now heading directly to the left at the same speed, they will all hit the left wall and bounce at the same time. This is an example of what is NOT allowed.

While it is technically possible to accomplish this without using arrays, it would mean typing much of the program 10 times. Don’t do that; use arrays.

**Extra credit** – Use random numbers to make the animation more interesting, so it doesn’t play the same way every time. The three clicks have to work the same way, but you could make the balls appear in random positions, so that each time the program runs, the balls start in different places. You could make the colors of the balls different. You could make the speeds and directions of the balls different each time. Any one of these three changes would be worth 10% extra credit. Only one extra credit modification will be counted per student, however.

Upload the completed program to this assignment link by 5 minutes before the start of class on the due date.

Just as a reminder of how each ball should move**, here is the first bouncing assignment.**

Create a program that displays a 400x500 screen with a white background. The screen has a red circle on it. When the user clicks once anywhere on the screen, the circle starts moving. When it hits any of the four edges of the screen, the circle “bounces”, changes direction so that it stays on the screen and keeps moving. Make sure that the circle eventually will hit all four walls (as long as the user lets it run at least 30 seconds.) Once the ball has started moving, a second click anywhere in the screen stops the motion. Then one more click will close the screen and end the program.

Bouncing inside a rectangular box is pretty easy to simulate. If you have separate x and y components of the motion, when the object hits a left or right wall, multiply the y component of motion by -1. When it hits the top or bottom, multiply the x component by -1.

Make sure that no part of the circle ever leaves the screen. It’s OK if the bounce happens when the circle is still a few pixels away from a wall, but do not let it happen after part of the circle has left the screen.