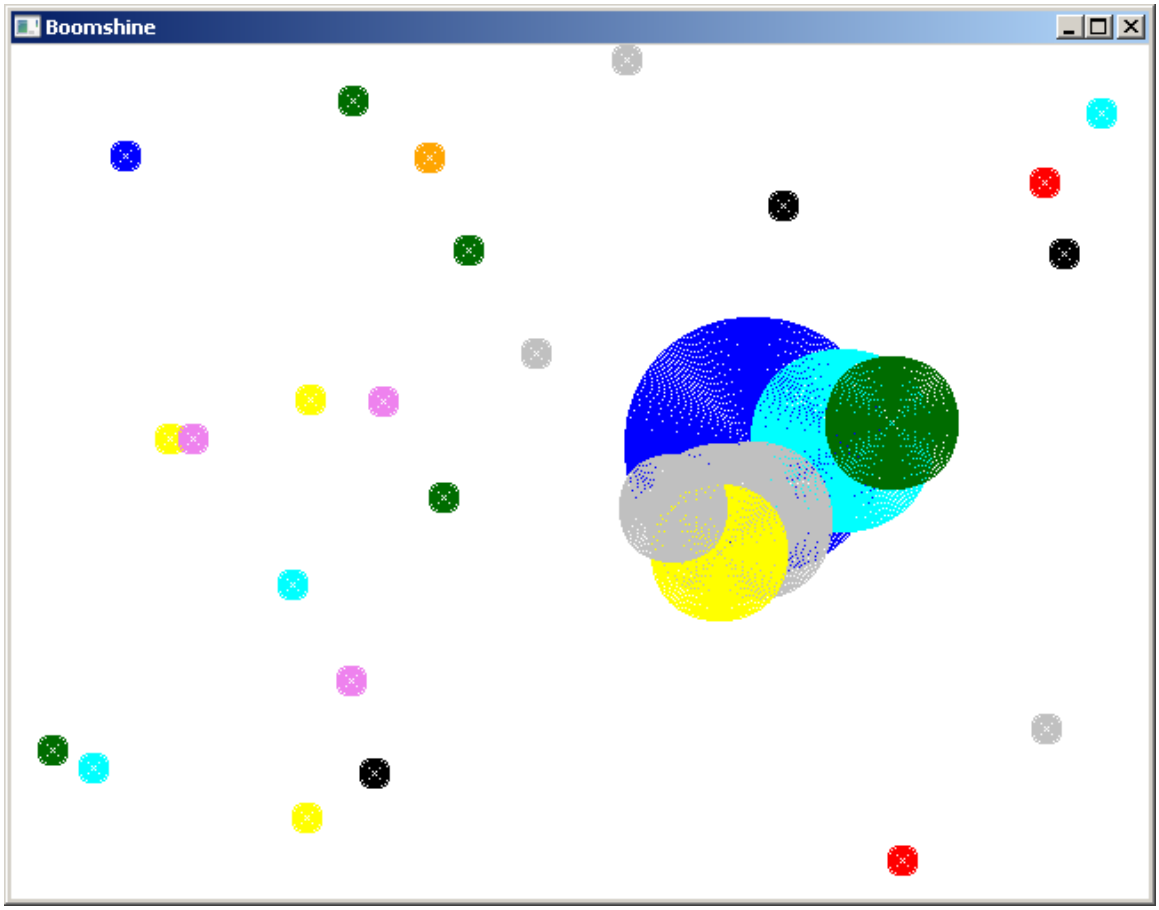
CS250 Assignment 6
Boomshine

Date assigned: Wednesday, April 9, 2014
Date due: Friday, April 25, 2014
Points: 60

Boomshine (http://www.addictinggames.com/strategy-games/boomshine.jsp) is a single-user game where a player tries to place a circle on the screen such that the circle causes the intersection of as many moving circles as possible. Once the initial expanding circle is placed on the screen, if a moving circle intersects the expanding circle on the screen, the moving circle becomes an expanding circle that doesn't move and begins to intersect any moving circles.

The first fixed circle is placed on the screen for a fixed period of time and expands during a given time period. After the time period expires, the circle disappears and can no longer intersect any of the moving circles. If all fixed circles use up their allotted period of time, the game stops and reports on the number of moving circles that were intersected.

You are to write a project called **Boomshine** that implements the game just described. I will place a video of our game in action on the schedule web page. Below is a screen shot of the game in action.



Here are the steps you need to go through to successfully complete this assignment.

Part I (Date Due: Wednesday, April 16, 2014 Points: 25)

1. In your CircleAnimation project, you are to implement the following interface for a Circle. I don't want you to mess with your existing Circle.h interface, that is why the class name is ACircle.h.

```
#ifndef ACIRCLE_H
#define ACIRCLE_H

#include "Color.h"

class ACircle
{

public:
  // x, y, radius, color
  ACircle (int = 10, int = 10, int = 10, Color = Color::AQUA);
  void setRadius (int);
  int getRadius () const;
  void setXCenter (int);
  int getXCenter () const;
  void setYCenter (int);
  int getYCenter () const;
  void setColor (const Color &);
  Color getColor () const;
  void draw () const;
  void drawFilled () const;
  bool intersectsWith (const ACircle &) const;

private:
  int mXCenter, mYCenter, mRadius;
  Color mColor;
};

#endif
```

2. Write the interface and implementation for MovingCircle in the project CircleAnimation which is a subclass of ACircle. A moving circle includes a speed and direction as additional attributes and behaviors that allow the setting/getting of the additional attributes as well as determining if a screen edge has been hit. If a screen edge has been hit, add a behavior that allows a circle to bounce off the screen edge.

3. Write a driver CircleAnimationDriver.cpp that places 25 moving circles with a random: a) location completely on the screen, b) color, c) direction, d) radius (5 to 25 inclusive), and e) speed of 1 on the screen. All circles are to bounce off the edge of the screen when the edge of the circle encounters the edge of the screen. The bouncing is to be natural, so for instance, a circle moving northeast and intersecting the right edge of the screen will change direction to northwest.

Part II (Date Due: Friday, April 25, 2014 Points: 35)

4. Write the interface and implementation for ExpandingCircle in the project CircleAnimation that is a subclass of ACircle. An expanding circle expands uniformly to a certain size over a given period of time. Add an additional attribute for the amount of expanding time.

5. Create a project called **Boomshine** in your existing solution.

6. Write the interface and implementation for Boomshine which plays the game of Boomshine as previously described. Here are a few more details that are to be implemented in the game of Boomshine:

(a) The constructor for Boomshine is to accept an integer that specifies the level of the game being played. The number of moving circles initially moving on the screen is five times the level.

(b) All moving circles start out with a radius of 8 pixels.

(c) For the game of Boomshine, circles can only move in a diagonal direction (NE, SE, NW, and SW). This is not true in Part I, only Part II.

(d) Wait for the user to place a single expanding circle anywhere on the screen. The initial expanding circle has a radius of 8 pixels and expands by 1 pixel every iteration through the game loop for exactly 2 seconds (120 frames at 60 fps).

(e) If a moving circle intersects with an expanding circle, the moving circle becomes an expanding circle and expands by 1 pixel every iteration through the game loop for 2 seconds.

(f) After all expanding circles have exhausted their time, the game stops and the results are displayed as shown above.

(g) You will need to use functions _itoa and strcat to display integer values on the screen.

7. Write the driver for Boomshine that creates a Boomshine object and uses the Boomshine functions to play the game of Boomshine.

To complete this assignment you must:

1. Create a project called **Boomshine** with the proper interfaces and implementations for playing the game as described above.

2. Type the solution (fully documented/commented) to the problem into your project. The hard copy must be placed on the instructor's desk by the time class starts on the day that each part is due. The hard copy must be printed in color,

double-sided, and stapled in the upper left corner. Printing order is: ACircle.cpp, MovingCircle.h, MovingCircle.cpp, CircleAnimationDriver.cpp for Part I and ACircle.cpp, MovingCircle.h, MovingCircle.cpp, ExpandingCircle.h, ExpandingCircle.cpp, Boomshine.cpp for Part II.

3. Once you are sure that the program works correctly, it is time to submit your solution. You do this by logging on to Turing and placing your complete solution folder with all working correctly in the proper CS250 Drop folder. Make sure that you copy your program folder and don't move the folder. If you move the folder, then you will not have your own copy!

As always, start early and see me if you have any design or implementation questions.

Extra Credit

The game of Boomshine shown in the link on the first page of notes has levels of play. I will give extra credit for anyone implementing levels of play as in the original game. That is, you are to implement the following from the original game including the PLAY again button.

From what I can tell, here are the number of circles displayed and the number of circles that must be intersected to move to the next level.

Level	Circles Displayed	Intersected Circles Necessary
1	5	1
2	10	2
3	15	3
4	20	4
5	25	5
6	30	10
7	35	15
8	40	21
9	45	27
10	+5 more	+6 more