CS250 Assignment 6: Boomshine

Date assigned: Friday, April 12, 2019
Date due: Friday, April 26, 2019
Points: 50

Goals for this assignment

1. Read and use existing code
2. Write an object-oriented program using multiple classes.
3. Use composition and inheritance.
4. Implement the Boomshine class and ExpandingCircle class.
5. Practice with basic 2D graphics in SDL.

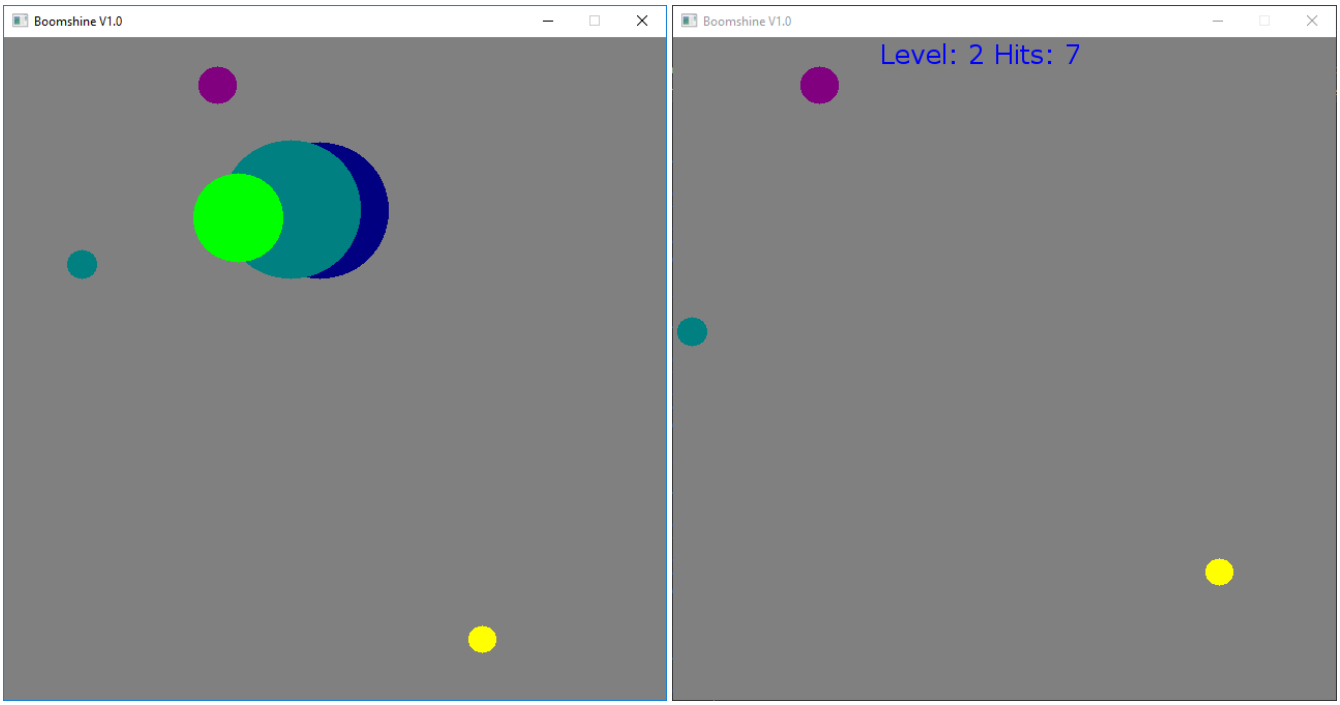
Boomshine Game

For this assignment, you will be writing a C++ / SDL program to implement the game Boomshine.

Boomshine (http://www.k2xl.com/games/boomshine) is a single-user game where a player places an expanding circle on a screen containing moving circles, such that the expanding circle collides with 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.

Expanding circles expand to a certain size and then disappear. The game stops once there are no more expanding circles on the screen. The game then reports on the number of moving circles that were intersected by an expanding circle.

You are to implement one level of the game Boomshine. Below is a screen shot of the game and you can find a video of the game on Grace.



Starting Point

Your starting point for this assignment is (06Boomshine-PUNetID), which is in the public folder on Grace. Copy this folder to your desktop and replace the word PUNetID in the project folder name with your own PUNet ID. The solution contains three projects:

a. **SDLManager:** contains the code that handles drawing to the screen. YOU MUST NOT MODIFY/CHANGE ANYTHING IN THIS PROJECT.
b. **Graphics2D:** is where you will write the code for **Expanding Circle**. This project contains the following:
    i. **Circle.h and Circle.cpp:** class that represents a single circle with x and y coordinates and a radius. You may need to add new functions.
    ii. **MovingCircle.h and MovingCircle.cpp:** class that is derived from Circle and represents a moving circle with a Direction, speed, and the bounding box of movements. You may need to add new functions.
    iii. **ExpandingCircle.h and ExpandingCircle.cpp:** class that is derived from Circle and represents an expanding circle. These files are currently empty.
    iv. **Direction2D.h** and **Direction2D.cpp**: class that represents the direction of movement. The directions are N, S, E, W, NE, NW, SE, SW. This class contains functions to get the x and y coordinates after a movement and functions to reflect off of all four sides of the window. DO NOT MODIFY.
    v. **Graphics2DDriver.cpp**: contains main, which sets up the window and handles the game loop.
    vi. **circles.txt:** text file containing the specifications for the circles. This file will have a maximum of 50 circles and will not contain any errors. The file will not end with a blank line.
c. **Boomshine:** is where you will write the code to implement the game Boomshine. The project must contain the following:
    i. **Boomshine.h and Boomshine.cpp**: will handle all of the gameplay for Boomshine.
    ii. **BoomshineDriver.cpp**: will contain main and the game loop. This file will contain an object of type Boomshine and use that object to handle the gameplay.

Build and run Graphics2D. It should display the moving circles to the screen.

<u>Step 1: Implement ExpandingCircle - Due on Monday 4/15:</u>

Write the interface and implementation for a new class **ExpandingCircle** (a subclass of Circle) in the project **Graphics2D**. An expanding circle's radius expands by one every time through the game loop up until it reaches 120. The initial radius for the expanding circle is 15.

Modify the Graphics2DDriver so that the user can click on the screen and a single expanding circle appears on the screen. The circle will expand until the radius is 120, then it will stop expanding. See the 06ExpandingCircle.mp4 movie on Grace.

<u>Step 2: Implement Boomshine – Due on Friday 4/26:</u>

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:

1. 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 (level can be 1 to 10 and the

default level is 1).

2. All moving circles start out at a random position on the screen (you need to ensure that the circles will appear within the screen), with a random radius of 10 to 19, a random direction (8 possible directions), a random speed of 1 to 3, and a random color between SILVER and FUCHSIA (14 possible colors). See line 24 in Color.h.

   As a reminder, in order to use rand(), you need to include both cTime and stdlib. You then need to seed the random number generator by typing this statement once at the top of main():

   ```
   srand(static_cast<unsigned> (time(static_cast<time_t *>(NULL))));
   ```

   You can then generate a random number using rand(). For example, to generate a random number between 1 and 3, you would type:

   ```
   int num = rand() % 3 + 1;
   ```

3. After the random circles have been generated, then you wait for the user to place a single expanding circle anywhere on the screen. The initial expanding circle has a radius of 15 pixels and expands by 1 pixel every iteration through the game loop until the radius is 120. The default color for the expanding circle is black. Once the expanded circle reaches 120, it should be removed from the screen.

4. 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 until the radius is 120. The color of the new expanding circles will be the same as the color they were when they were moving circles. Again, the expanding circles will disappear once their radiuses reach 120.

5. Once all expanding circles have disappeared, the game stops and the results (level, and number of moving circles that were intercepted by expanding circles) are displayed as shown above. The text is displayed using 25-pt verdana.

6. You will need to use the function to_string() to display integer values on the screen. Look this function up.

7. You will need to handle the collision/intersection of circles. Add a function to the Circle class that determines if a circle that has been passed in as a parameter intersects/collides with the circle calling the function. Collision is determined by calculating the distance between the two points (centers of the circles) and checking if the distance is less than or equal to the sum of the two radiuses.

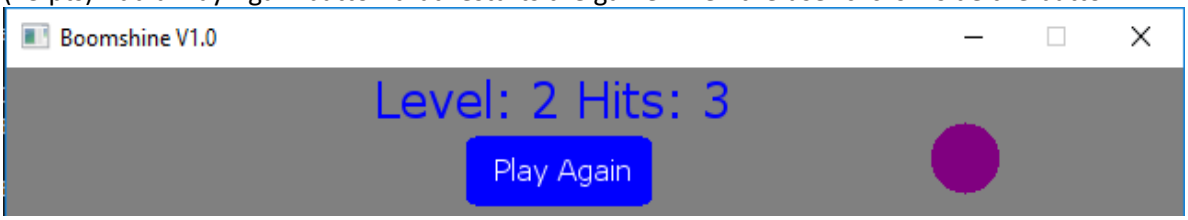
Write the driver for Boomshine that creates a Boomshine object and uses the Boomshine functions to play the game of Boomshine.

1. Allow the user to close the window using the x icon in the top right of the window.
2. Your game should handle multiple levels, but the version that you submit must use level 2. In other words, pass in 2 to the constructor for your Boomshine object.

Bonus

1. (+ 2 pts) Add a blast sound each time a moving circle collides with an expanding circle. Only one blast is to sound for each collision.

2. (+3 pts) Add a Play Again button that restarts the game when the user clicks inside the button.



Your project must be on time (both the electronic and hard copy) to receive any bonus points.

To complete this assignment, you must submit the following:

1. An electronic copy of your program on Grace

a) You need to follow the coding standards from the CS250 Web page Make sure that you include the hours you worked on the assignment in your header comments.
b) Pay attention to the example output. Your program's output must look **exactly** like the sample output.
c) Make sure that your program builds without errors & warnings and runs correctly. If you get any errors or warnings, double check that you typed everything correctly. Be aware that C++ is case-sensitive. You will lose 10% if there are any warnings and 40% if your program does not build successfully.
d) Once you are sure that the program works, it is time to submit your program. You do this by logging on to Grace and placing your complete solution folder in the **CS250-0X Drop** folder.
e) The solution must be in the drop folder by the time class starts on the day the assignment is due. Anything submitted after that will be considered late.

2. A hard copy of your program

The hard copy must be placed on the instructor's desk by the time class starts on the day that it is due. Print BoomshineDriver.cpp, Boomshine.h, Boomshine.cpp, ExpandingCircle.h and ExpandingCircle.cpp in that exact order.

a) **The hard copy must be printed in color, double-sided, and stapled in the upper left corner if your solution contains multiple pages**. Failure to print properly will result in loss of 4 points (10%)
b) Your tab size must be set to 2 and you must not go past column 80 in your output.

Remember, if you have any problems, come to me straight away with your project on Grace. Good Luck!!!! :)