CSCI 200 - Fall 2023
Foundational Programming Concepts & Design

A4 - SFML: Bubble Bobble

→This assignment is due by Tuesday, October 31, 2023, 11:59 PM.←

→ As with all assignments, this must be an individual effort and cannot be pair programmed. Any debugging assistance must follow the course collaboration policy and be cited in the comment header block for the assignment.←

→ Do not forget to complete the following labs with this set: L4A, L4B, L4C ←

→ Do not forget to complete zyBooks Assignment 4 for this set.←

Jump To: **Rubric Submission**

Concepts

You are going to create magic bouncing balls to watch as they move around the screen.

Class Creation

For this assignment you must first create a class called `Bubble`. The `Bubble` class needs to have the following private data members to store information:

- a `CircleShape` which will be used to draw the Bubble

RMP-H

- two `double`s to store the X and Y direction respectively the Bubble should move (call them `xDir` and `yDir`)

Be sure to have the necessary getters and setters for your private data members and/or you may make non-default constructors as you deem necessary to set initial values on your `Bubble` class.

Draw It!

In our `main()`, before we get to the draw loop we want to make our `Bubble` object. We want only one instance to exist in memory. If we made the object inside the draw loop, then every iteration of the draw loop a new object would be created.

Next inside our draw loop, after we've cleared the window and before we display the window, we need to draw the bubble. Add a public method to the `Bubble` class called `draw()` that accepts an SFML `RenderWindow` object by reference. The function then draws the `CircleShape` member of the class.

At this point, run your program and you should see a white circle. Perfect! Let's continue.

Move It!

We need to add another public member function called `updatePosition()`. This function needs to add the direction values to the position of the `CircleShape`.

When we create our `Bubble` object, we need to make sure its direction values are set to non-zero values. Do this now, we suggest using values in the range of `[0, 0.2]` for xDir & yDir respectively. (Note: feel free to tweak these values up/down if it moves too slow/fast.)

Now in our draw loop, after we've checked the events we will check the time. If the elapsed time has eclipsed 1/60th of a second, then call `updatePosition()` on the object.

Run your program. It should be moving now. Oh no! It went off the screen. Let's keep it in view.

Bop It!

The last part of the bubble is have it bounce around the window. Recall that the position for a `CircleShape` corresponds to the upper left corner of a square enclosing the circle. We need to check if any of the edges of our circle move outside the window. We can check this by seeing if

the position becomes less than zero (to correspond to the left or top of our window) OR if the position plus the diameter is greater than the width or height of our window (to correspond to the right or bottom of our window). If any of these conditions occur, then we simply need to reverse the corresponding direction of our bubble to simulate a bounce. Once we've checked all four sides, we're contained!

Modify `Bubble::updatePosition()` so that it accepts the window width and height as parameters. Now after moving the bubble, check if the bubble went over a wall. If it did, move the bubble backwards and then reverse the necessary direction.

We recommend to start with the right wall first, then bottom, then left, then top. Get them working one at a time and keep adding to your code.

Run the program and watch it stay within the window. Excellent.

Bubbles Bubbles Everywhere

We want to replace our single `Bubble` object with a `vector` of `Bubble`s. Before your draw loop, create a `vector` of five bubbles initially. Give each bubble a random starting position between `100` and `400` for X and Y and a random direction for X and Y in the range `[-0.1667, 0.1667]`. Additionally, give each Bubble a random radius between 10 and 50. Lastly, give each Bubble a random color so we can tell them apart.

Inside our draw loop, we now need to draw all the Bubbles in our vector. After our event handling, we'll then need to update the positions of all the Bubbles in our vector. You should now see five Bubbles bouncing around the window. Excellent. Let's have the user interact.

Pop-A-Bubble

When the user clicks the left mouse button, we want to create a new bubble positioned at the location where the user clicked. This new bubble should have the same random starting properties that our original bubbles did. After the user clicks the first time, we should see six bubbles moving around the window. A second click, seven bubbles. And so forth as the user continues to click.

We may get to the point where there are too many Bubbles on the screen. If the user presses the D key, then we want to delete the last bubble that was added to the window. Obviously if there are no Bubbles in the window, then pressing D should do nothing.

If the user starts your program, clicks twice, then presses D three times, we should be seeing four bubbles on the screen.

UI Design

As a final step to make the program more user friendly, if the user presses the Q or Escape key, automatically close the window.

Grading Rubric

Your submission will be graded according to the following rubric:

Points	Requirement Description
0.5	Submitted correctly by Tuesday, October 31, 2023, 11:59 PM
0.5	Project builds without errors nor warnings.
2.0	**Best Practices** and **Style Guide** followed.
0.5	Program follows specified user I/O flow.
0.5	Public and private tests successfully passed.
2.0	Fully meets specifications.
6.00	**Total Points**

Submission

Always, **always**, **ALWAYS** update the header comments at the top of your main.cpp file. And if you ever get stuck, remember that there is LOTS of **help** available.

Zip together your `main.cpp, Makefile, *.h, *.hpp, *.cpp` files and name the zip file `A4.zip`. Upload this zip file to Canvas under A4.

→This assignment is due by Tuesday, October 31, 2023, 11:59 PM.←

→ As with all assignments, this must be an individual effort and cannot be pair programmed. Any debugging assistance must follow the course collaboration policy and be cited in the comment header block for the assignment.←

→ Do not forget to complete the following labs with this set: L4A, L4B, L4C ←

→ Do not forget to complete zyBooks Assignment 4 for this set.←

Last Updated: 05/26/23 20:12

Any questions, comments, corrections, or request for use please contact jpaone {at} mines {dot} edu.

[Jump to Top] [Site Map]