Exercise 2
Class and Object Manipulations

Overview

- This exercise is to be conducted **outside of the class.**
- You will be adopting a **Pair Programming** strategy in doing this exercise.
  What is pair programming? (https://youtu.be/oBraLLybGDA)
- You can maintain your partner from the previous exercise or change to a new one.
- You and your partner will be coding collaboratively online using VS Code and Live Share.
  Using Live Share for online collaborative coding (https://youtu.be/s9hfONtUcR8)
- You will communicate to each other using Google Meet, Webex or any online meeting tool.
- You will record your pair programming session.

Pair Programming and Collaborative Coding

- Pick any time worth **TWO (2) hours** (maximum) within the given date to conduct the pair programing session with your partner.
- You may also split your pair programming into several sub-sessions provided the total time is still within 2 hours.
- Log the date and time for every pair programming session conducted. Write them in the program source code.
- Record the meeting about your pair programming session. If you do your programming in multiple sessions, record all of them. You do not have to edit the video.
- Code submissions without the video at all or the video was too short, will be declined.

Notes:

- You are advised to explore the exercise on your own first before doing the pair programming session with your partner. This should make yourself be more prepared for the pair programming session.

How To Record the Session

- You can choose Google meet as your online meeting tool and use the feature **"record meeting"** to record your pair programming session.
- Note that a free personal google account does not support the recording feature.
- Try using your student or graduate account from UTM to be able to access the "record meeting" feature.
- Alternatively, you can record your pair programming session locally using software like OBS, PowerPoint, etc. https://elearning.utm.my/21221/mod/page/view.php?id=26194

About the Video

- The video must show that you are coding, communicating, and collaborating with your partner. In this regard, **speak only in English**.
- In the video you should show your VS Code and the output (console).
- You can record the session in a single or multiple videos. If you use multiple videos, put them in a folder, and submit only the folder's link.
- Set the video file (or folder) permissions with "Restricted" and add my account jumail@utm.my to allow me to access it.
- Make sure the video is available until the end of the semester.
- Submit the raw videos, i.e., you don't have to do post-editing.

Notes:

- Please make the font-size of your VS Code a little bit larger so that it easy for me to see your code in the video. You can do this by pressing the key **Ctrl** and **+** in VS Code.

Plagiarism Warning

You may discuss with others and refer to any resources. However, any kind of plagiarism will lead to your submission being dismissed. No appeal will be entertained at all.

Late Submission and Penalties

- The submission must be done via eLearning. Other than (such as telegram, email, google drive, etc.), it will not be entertained at all.
- Programs that CANNOT COMPILE will get a 50% penalty.
- Programs that are submitted late will get a 10% penalty for every 10 minutes.

Question

In this exercise, you will be writing a C++ program that displays a list of circles moving around on the screen. You will be using an Object-Oriented Programming (OOP) approach to write the program. You will write the program in a single source code file, but the class specification (declaration) and implementation (definition) are separated.

You are provided with a codebase containing class `Circle` with some parts of the class have been given as follows:

- `x` and `y` – attributes to hold the center location of the circle.
- `radius` – attributes to hold the size of the circle
- `dx` and `dy` – attributes to hold the movement of the circle in horizontal and vertical direction, respectively. The magnitude value represent the speed, whereas the sign (- or +) represents the direction. For example, if `dx` is -10, it means the circle moves to the left with the speed of 10, while `dx` = 10 means, it moves to the right with the speed of 10.
- `Circle()` - two constructors are provided in the codebase, a generic and default constructor.
- `setLocation()` - a mutator method for the center location attributes, `x` and `y`.
- `setMovement()` - a mutator method for the movement attributes, `dx` and `dy`.
- `draw()` and `undraw()` – methods to show and clear the circle on / from the screen, respectively.
- `move()` – a method that updates the location of the circle (`x`, `y`) to mimic the movement.

Modify the codebase (**exercise.cpp**) to achieve the goal of the program. Do the following tasks:

1. Define two more constructors below:
    a. An overload constructor that accepts the center location and the radius of the circle, whereas the movement are set to zero. Also, the radius should have a default value, for example 100.
    b. A copy constructor that copies only the center and radius of another circle, and sets the movement attributes to zero.

2. Define an operation to determine the distance between two circles using three approaches below.
    a. with a method (or member functions)
    b. with an overload minus (-) operator
    c. with a friend function. As for this approach, let the function accesses directly the class attributes.

    See the Formulas section to learn about how to determine the distance.

3. In the main function,
    a. Declare an array to hold a list of `Circle` objects. Initialize the array with five circles as follows: Four circles are located at each corner and one at the middle of the screen.
    b. Show all the circles on the screen.

    See the video for the expected output of this task.

As for Task 4 to 6 below, you may add additional methods whenever necessary. See the video for expected output of each task.

4. Add the following features to the program that will be interacted by the user using the keyboard:
    a. When the key 'R' is pressed, each circle is re-located to a random location. The circle should be fully inside the screen.
    b. When the key 'M' is pressed, each circle moves. The speed and direction are randomly set.
    c. When the space bar key is pressed, each circle stops moving.

5. Add the following features to the program that will be interacted by the user using the mouse:
    a. When the mouse is clicked on an idle circle, that circle will move. The speed and direction are randomly set.
    b. When the mouse is clicked on a moving circle, that circle will stop moving.

    As for this task you will need two add two methods below to the class:

    - `isMouseClicked()` which determines whether the circle is clicked on by the mouse. This method accepts the mouse coordinates as parameters and returns **`true`** if the mouse clicks on the circle. See the Formulas section to learn about how to determine whether the mouse is in a circle.
    - `isMoving()` which determines whether the circle is moving.

6. Extend the program to simulate bouncing effects that occur in two scenarios below:
    a. Circles reach the screen boundaries.
    b. Circles hit each other.

    As for this task you will need two add the following methods and operator

- `left()`, `right()`, `top()` and `bottom()` – helper methods which return the edge the circle, respectively. You will use these methods in Task 6(a).
- an overload operator == which will be used for operations like

  `c1 == c2`

  where **`c1`** and **`c2`** are both of type `Circle`. This operator returns **`true`** if the two circles **overlap**, and false if otherwise. See the Formulas section to learn about how to determine whether two circles overlap. Use any operation that calculates the distance from (2) to implement this operator.

Formulas

1. Distance between two circles:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

where (x_1, y_1) and (x_2, y_2) are the coordinates of the center points for the two circles respectively.

2. Determining whether two circles overlap:
   Two circles overlap if the distance between them is less than or equal to the sum of their radius. The following figures illustrate this scenario.

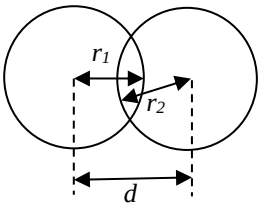

Figure (a): Overlap

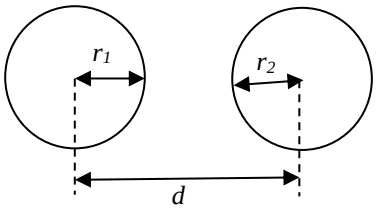

Figure (b): Do not overlap

where r_1 and r_2 are the radius of two circles respectively, and d is the distance between the circles from their centers. An overlap occurred in Figure (a) as $d \leq (r_1 + r_2)$, while there is no overlap in Figure (b) as $d > (r_1 + r_2)$.

3. Determining whether a mouse clicks on a circle:
   Consider a mouse location as a circle with the radius of 0. Thus, you can make use of the formula about circles overlap testing above to determine whether a mouse is inside a circle.

Resources

1. WinBGIm documentation

   https://home.cs.colorado.edu/~main/cs1300/doc/bgi/

   See the following functions. You may need using them in this exercise.

   - `circle()` – to draw an outlined circle
   - `fillellipse()` – to draw a filled or shaded ellipse.
   - `setcolor()` – to specify the outline color
   - `setfillstyle()` – to specify the fill or shade color and pattern
   - `delay()` – to pause program for certain time
   - `getch()` – to read a character from the keyboard
   - `kbhit()` – to test whether a user presses any key
   - `getmouseclick()` – to read input from the mouse click

2. How to use random numbers in C++

   https://www.cplusplus.com/reference/cstdlib/rand/

Assessment

This exercise carries **3%** weightage for the final grade of this course. The breakdown weightage is as follows (out of 100 points):

Criteria	Points
1. The code a. Task 1 –Constructors b. Task 2 – Calculate distance i. Method ii. Overload operator iii. Friend function c. Task 3 – Create and show an array of circles d. Task 4 – Features controlled with the keyboard i. Random location ii. Moving iii. Stop moving e. Task 5 – Features controlled with the mouse i. Click on idle circles ii. Click on moving circles f. Task 6 – Bouncing effects i. Circles reach the screen boundaries ii. Circles hit each other	 10 10 10 15 15 20
2. Pair Programming Session a. Video and overall b. Active collaboration c. Both members play both roles Driver and Navigator.	 10 5 5

Submission

- Deadline: Saturday, 4 Dec 2021, 11:59 PM
- Only one member from each pair needs to do the submission.
- Submission must be done on eLearning. Any other means such as email, telegram, google drive will not be accepted at all.
- You will need to submit TWO (2) items:
    a. The source code: submit only the source code, i.e., **exercise.cpp** file.
    b. The video link. Submit the link via eLearning as well.

FAQs

1. **Who will be my partner?**

   Choose your own partner. You may maintain your partner from the previous exercise.

2. **Can I pair up with someone from a different section?**

   No.

3. **Can I do the exercise alone?**

   This is only allowed if the number of students in the class is imbalanced. You also need to ask for permission from the lecturer.

4. **What do we need to show in the video?**

   You should show that you are **doing pair programming** rather than explaining about your code. The video is not meant for presentation.

5. **Do we need to switch roles between Driver and Navigator?**

   Yes. Your video should show that you and your partner keep switching between these two roles. No one should be dominant or play only one role.

6. **What if I do this exercise alone? Do I still need to submit the video?**

   In case you got permission to do the exercise alone, you still need to submit the video. You show in the video your progress in doing the exercise. You need to talk about what you are currently coding.

7. **What if we do pair programming physically (face to face).**

   You and your partner should use only one computer and sit side-by-side. You do not have to open LiveShare and online meetings. You can record the video locally using software like OBS. Again, you still need to talk and discuss with your partner in the video. It is also recommended to turn on the web camera. Keep in mind that you keep following the SOP about COVID-19 when working in a face-to-face environment.

