CS 101: Computer Programming and Utilization, Autumn 2021

Bubble Trouble

Introduction

In this project you will use C++ to create a game, Bubble Trouble. Your aim is to clear all the bubbles. Destroy the bouncing bubbles by splitting them using bullets, but don't let them touch you! Are you up for the challenge?
To help you get started, we have provided a starter code which implements basic functionality of the game. Your task is to complete the game by implementing all the required features and further enhance it by implementing at least 2 extra features.
A screen recording of the functionality provided in the starter code, required features and extra features can be found at
https://drive.google.com/drive/folders/1kDKONIZFMCGAQDXFlkHTXkvsipljLYTF?usp=sharing.
These are provided as a reference.

- `starter_code_features.mp4` demonstrates the features already available in the starter code given to you.
- `required_features_win.mp4` and `required_features_lose.mp4` demonstrates the required features to be implemented. `required_features_win.mp4` represents a scenario in which the user wins, while `required_features_lose.mp4` represents a scenario in which the user loses.
- `extra_features.mp4` demonstrates all the extra features that can be implemented.

Starter code

The starter code can be found at https://drive.google.com/file/d/1vMY0LrRnwn5doVO-2EFu6l4rTukSLXSN/view?usp=sharing.
You can compile the starter code using
```
s++ main.cpp
```
and run the program using
```
./a.out
```
Specifically, the starter code consists of the following files.

`bubble.h`

This file implements the class `Bubble`, which uses `Circle` from simplecpp graphics to represent the bubble in the game. The position of the bubble updates at every time step based on its velocity. We manually initialize the position and velocity of multiple bubbles. The bubble bounces against the two vertical borders of the window.

`bullet.h`

This file implements the class `Bullet`, which uses `Rectangle` from simplecpp graphics to represent the bullet in the game. The position of the bullet updates at every time step based on its velocity.

`shooter.h`

This file implements the class `Shooter`, which uses shapes from simplecpp graphics to represent the shooter in the game. In this case, we use a circle and a rectangle to represent the head and body of the shooter. Based on user input, the shooter can move left/right and also shoot bullets.

`main.cpp`

This file consists of the game simulation loop implemented using events. We use the following keys for user interaction

- a: move shooter left
- d: move shooter right
- w: shoot 1 bullet
- q: quit

The main function keeps track of the bullets and bubbles in the game using a `vector`.

Functionality in starter code

The starter code provides functionality to

- Create a window with a shooter and some bullets
- Horizontally moving bubbles bouncing against the vertical walls
- Ability to shoot a bullet from shooter vertically upwards

Project Task

The project will be considered successful only if you

- complete the basic version of the game by implementing the required features, and
- enhance the game experience by implementing at least 2 extra features

You have to ensure that the game runs at interactive frame rate, i.e. there is no visible delay between the user's input and the movement of the graphical elements.

Required Features

The following features are necessary to complete the basic version of the game.

- Modify the bubble movement to incorporate vertical motion such that the bubbles travel in a parabolic path. When the ball hits the horizontal ground, it should bounce off the ground.
- Introduce collision between the bubble and the bullet; the bubble disappears after a bullet hits it. After a collision, you can make the bullet disappear too.
- Introduce collision between the bubble and the shooter; the shooter is dead if a bubble touches it.

Extra Features

You are supposed to enhance the game experience by implementing at least 2 extra features from the following

- Use bubbles of different sizes. When a bullet hits the larger bubbles, they should split into smaller bubbles (half the radius of the original larger bubble) and move in horizontally opposite directions.
- Add a score, time and a health counter. The score increases whenever you hit a bubble. The game gets over when we run out of time (measured in seconds) or if we run out of health (whenever the bubble hits the shooter, the health reduces by 1).
- You can implement different levels of the game. Once you destroy all the bubbles in one level, you can move to the next level which can be difficult in terms of more bubbles, faster moving bubbles, less time and/or more obstacles for bouncing. The bubbles in different levels must have different colors.

Project Rules

- The project is to be done individually. You are not allowed to discuss with anyone including friends/family, no code sharing is allowed. No help from TAs as well. (TAs will clarify concepts as related to the course).
- The code will be subject to plagiarism checking.
- The project will be graded by taking individual vivas.
- The deadline for submitting the project is **10:30pm Sunday, Feb 27 2022**. Vivas will be held during the week **from Feb 28 2022 to March 5 2022**.
- The project has two grades: *pass* / *nopass*
- The project will be graded as *pass* based on the following

  - program runs successfully demonstrating all the required features and at least 2 extra features, and
  - code follows good programming practice (explanatory variable names, indentation, comments)
  - you should be able to explain all aspects of your code during the viva

- The only way to obtain AA grade in the course is

  - get AB in normal evaluation, and
  - get a *pass* in the project

- A *nopass* in the project means no change in grade obtained otherwise.
- The project **cannot** be used for any other kind of grade change (e.g., you cannot use the project to upgrade a BB to AB and so on), nor can it be used in lieu of any other evaluation.
- Instructor's decision about the project grade will be final.
- No extensions to the deadline will be given for any reason (including medical).
- Project code submission details and requirements are given in the document linked below. The submission will be on BodhiTree and link for the submission will be made available in the first week of February, 2022.

Submission Instructions

- You are required to create the following as part of your submission

  1. Code files (.cpp, .h)
     Your program should be able to compile using the command `s++ main.cpp`
  2. Video (.mp4)
     Do a screen recording of your Graphics window demonstrating the features you have implemented. Upload it to your Google drive, create a shareable link and provide the link in the `README.txt` helper document. We will check that the timestamp of the video file uploaded is before the submission deadline. You can use any screen recorder (e.g. VLC, OBS) to record your window.
  3. Helper document (`README.txt`)
     This is a text file (.txt file) that lists the features that you have implemented. Do not forget to include the link to the video in this file.

- Add the code files and the helper document in a folder named `rollnumber` where rollnumber is your roll number and create a .zip file named `rollnumber.zip`.
  For example, if your roll number is 210512345, then your folder should be named `210512345` and the zip file should be named `210512345.zip`
  Do **not** include the video in your zip file. Your folder structure should look like this

```
210512345
|--bubble.h
|--bullet.h
|--main.cpp
|--shooter.h
|--README.txt
```

- The zip file has to be submitted on BodhiTree.
- A link will be made available for submission on BodhiTree in the Programming Labs section in the first week of February 2022.
- The deadline for submitting the project is **10:30pm Sunday, Feb 27 2022**.