CGRA151 2019 Assignment 2: bouncing ball and straight-line drawing

Learning objectives

Core assignment: understand basic physical simulation: how to make an object move, how to handle collisions, experience of simple vector mathematics.
Completion assignment: be able to efficiently manage multiple instances of an object.
Extension assignment: demonstrate an ability to implement an algorithm to draw a straight line.

Practical matters

You should submit up to three sketches: one for core, one for completion, and one for extension. Each sketch will be in its own directory. You will need to use the zip utility to put all the directories into a single zip file and upload that one zip file. You then upload that one zip file to the ECS submission system. For example, if you complete all four sketches, and give them the names you are told to give them, then you can run this command to make the zip file:

```
zip -r A2submission.zip A2Core A2Comp A2Extn
```

The "`-r`" option tells zip to include all of the files inside the directories. The file that you submit is A2submission.zip. See the course website for details of how to use the submission system.

Your sketches must run on the version of Processing used on the ECS machines without needing to be modified.

Marking

Core assignment: completing this satisfactorily can provide up to **50%** of the marks.
Completion assignment: completing this will provide a further **30%** of the marks.
Extension assignment: completing this will provide a further **10%** of marks.
Code quality: the final **10%** of the marks will be awarded based on the quality of your code.

Core assignment — bat and ball

Implement a bouncing ball in 2D. The ball should be drawn as a filled circle. It should bounce off the sides of the window. The ball should start near the top left-hand corner of the window and move with a velocity that includes both x and y components so that we can see the ball bounce off all four sides of the window.

Add a bat. For the purposes of the core assignment, the bat is an axis-aligned rectangle that is controlled by the mouse. Axis-aligned means that the rectangle's edges are parallel to the edges of the window.

Whenever the ball hits the bat it must bounce off the bat in a plausible manner, but it does not need to be physically accurate at this stage.

Hint: For bouncing off the window's sides, you need to consider four cases: top, bottom, left, and right. For bouncing off the bat, you should consider the same four cases. This might lead to odd behaviour when the ball bounces off a corner of the bat, but that is OK for the core assignment.

For full credit, your simulation should be *reasonably* stable: the ball should not *noticeably* lose energy or gain energy. In practical terms, that means that it should look stable over the course of, say, one minute.

Save this sketch as `A2Core`.

Completion assignment — turning it into a game

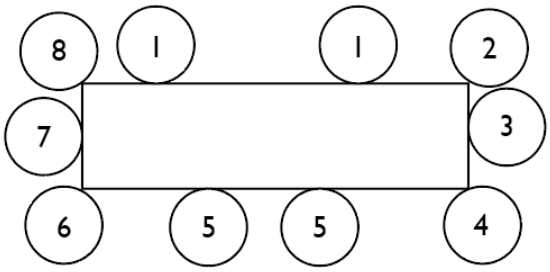
Save a fresh copy of your core sketch, as `A2Comp`, and modify it as follows.

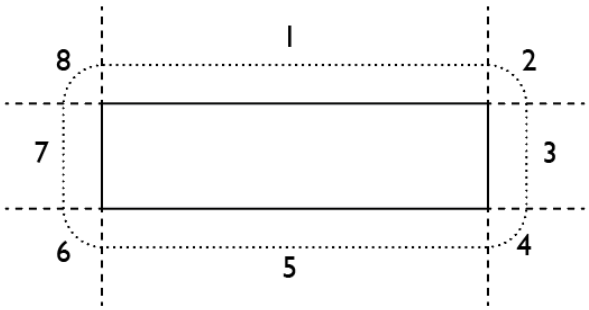
Add a number of rectangular obstacles that the ball bounces off. You should be able to reuse the code you wrote for the bat to handle bouncing off the obstacles. You will need to think carefully about how you store the rectangles: you need to store position, size, and status. Rather than using multiple arrays/lists for these different attributes, you should consider making a class and store a single array/list of objects of that class.

You now want to turn this into a (simple) game. Each rectangle will have a colour that indicates its status: green = never hit, yellow = hit once, red = hit twice. When a rectangle has been hit three times it vanishes. The aim of the game is to remove all the rectangles.

Just to be clear: you still need a bat rectangle, which follows the mouse and which does not change colour or vanish. The final result is a simplification of the video game "breakout".

Finally, improve the ball-rectangle interaction by considering eight cases (instead of four): one for each side and one for each corner. For full credit the interaction with the corners should be reasonable realistic, but does not have to take into account accurate angles of reflection.





Save this sketch as `A2Comp`.

Extension assignment — straight-line drawing

In the lectures you are shown how a computer graphics algorithm can draw a straight line by drawing the correct set of pixels. Implement a straight-line algorithm and implement a way that you can clearly demonstrate to your marker that it works for a range of start and end points. See slides 71-73 in the lecture notes for how you are expected to do this.

Save this sketch as `A2Extn`.