Examiners' coursework report 2015–16

CO1112 Creative computing 1 – Coursework assignment 2

Introduction

This coursework assignment explored the use of 2D motion, acceleration under gravity, collision detection, and keyboard interaction. It also gave students an opportunity to develop their creative skills.

Most students performed very well on this coursework, with over 75 per cent of submissions being graded as **excellent** or **very good**.

On the other hand, a small number of students failed the coursework assignment. In all such cases, this was due to submitting incomplete work; in particular, all but one of the failed submissions did not include an attempt at the final question (Part 6) at all. Part 6 was worth 40 per cent of the overall mark for the coursework assignment, so any submission that did not even attempt this question faced an uphill battle to achieve even a pass grade.

As with all courseworks (and examinations), it is very important to consider the relative marks awarded for each question, and to set aside sufficient time to attempt the questions which are worth the most marks.

General remarks

The instructions for the coursework assignment stated that students should submit a short written description for each part of the work, in addition to their code, which should include a discussion of any problems encountered and any attempts made to solve them (whether successful or otherwise). Some students did not submit these descriptions, and therefore lost out on some fairly easy marks.

Regarding the code submitted, a common problem was lack of comments in the code. Some of the best submissions included not just neat, well-commented code, but also featured object-oriented design (this was not necessary, but could make the code more modular, flexible and extendible if done well).

For this specific coursework assignment, the logical order in which various steps were performed in the code was particularly important. A number of submissions had bugs associated with this. For example, some students applied collision detection tests before applying gravity to the ball, which could lead to situations where the vertical velocity of the ball had been reversed due to a collision with the floor, but then gravity was added to the velocity which re-reversed its direction. This demonstrates that even seemingly simple programming tasks such as this can often hold hidden complexities, and can require careful thought and testing to get right.

Comments on specific questions

Part 1

Most students had no problem with this part, and scored very highly. The most common problem was the use of hard-coded values in the code instead of variables – it is much easier to change the behaviour of the program if all values (e.g. gravity, x and y position and velocity, etc.) are stored in variables and initialised in one place. Another problem seen in some submissions was to place one-off operations (e.g. setting the frame rate) in the draw() method rather than in setup(). Remember that the setup() method just gets called once at the start of the sketch, but the draw() method gets called many times per second, so it is inefficient to place code in draw() that only needs to be called once.

Part 2

Again, the majority of students found this part easy and scored highly on it. The most common problem was to have the collision detection looking at the centre of the ball (i.e. its x, y position) without taking into account the ball's radius. For example, to check whether the ball has hit the left edge of the screen, the correct condition (assuming the ball's x position is stored in the variable x and its diameter in variable d) would be (x – d/2 <= 0).

Some students implemented the collision detection correctly, but had problems because they started the sketch with the ball at position (0, 0). In that case, the ball could immediately trigger the collision detection in the x and y directions (whether it actually did this depends on the order of applying various operations in the code). A simple fix was to move the start position of the ball so that it was a little away from the screen edges.

Part 3

Once again, most students completed this part successfully and scored high marks. It should have been fairly straightforward after consulting the suggested background reading.

One point to note is that the ball should bounce right up to its starting height on each bounce, unless there was a deliberate attempt to implement an inelastic collision (i.e. a loss of energy each time the ball hits the ground). Some students did implement inelastic collisions (e.g. by multiplying the magnitude of the vertical velocity by a factor less than 1.0 at each collision), and they wrote about this in their written description – these students generally scored very highly. Other students just implemented perfectly elastic collisions (no loss of energy), and this was fine too. However, a few students said they thought the elastic collisions looked unrealistic (which is a good observation) but then tried to fix this in an unprincipled way (e.g. by subtracting a constant value to the velocity at each collision regardless of magnitude or sign of the velocity, rather than using a multiplying factor) – these approaches generally ran into problems. The background reading should have given various hints on how to do this properly.

Part 4

Most students did a reasonable job here, although this part was a little more complex than preceding parts. A number of students tried to solve the problem with one conditional statement, whereas two separate checks were really required: one to check for collisions against the top or bottom

of the block (i.e. collisions in the y direction), and another to check for collisions against the left or right edges of the block (i.e. collisions in the x direction). One student actually implemented collision detection by checking for the colour of neighbouring positions rather than checking positions of other objects directly – this was an unorthodox approach, but worked perfectly well in this simple situation.

An added complexity in this part was that it involved a collision of a relatively fast moving ball with a relatively thin block. Various bugs were seen in the submissions, such as the ball passing right through the block, or getting 'caught' inside the block and continually triggering opposing collision conditions. The precise nature of the problems observed depended once again on the order in which collision and motion-related operations were applied. It was nice to see a variety of successful approaches applied to solving these kinds of problems, including adding some extra "padding" around the ball in the collision checks, or specifically resetting the position of the ball back to the boundary of the block once a collision had been detected. Students who discussed such issues in their written description generally scored highly.

Part 5

This part was generally done well. There is plenty of information available in the CO1112 subject guide, and on the *Processing* website, on how to handle keyboard interaction. One student pointed out that the choice of 'z' and 'x' keys to move left and right, as specified in the question, made sense on a QWERTY keyboard, but not on some other keyboard layouts, and went on to discuss internationalisation issues. This was a good point, and it is nice to see some students engage in work at this level.

Part 6

After some background reading, Parts 1–5 should have been fairly straightforward. In contrast, Part 6 was looking for more imagination and creativity. The submissions included some really outstanding pieces of work here, but also a whole spectrum of other standards, right down to some in which very little effort was apparent. Part 6 was worth **40 per cent of the whole coursework assignment**, and time should have been allocated to this part accordingly.

The brief for Part 6 involved developing a sketch that had a Rio Olympics theme, and which involved motion and collision detection techniques (along with other techniques encountered during the course). Some submissions did not successfully fulfil this brief; for example, there was no discernible collision detection involved; or indeed there was no Olympics-related aspect to the work. Such submissions struggled to gain many marks. On the other hand, some submissions showed a very imaginative and successful integration of all of these features, which was very pleasing to see.

As with all parts, students were asked to submit a written description of their work for Part 6, describing the ideas behind the design, the implementation, the problems faced; and, finally, a critical evaluation of the end result. Even some of the students who submitted very good sketches did not do a very good job at the write-up. In particular, the critical evaluation aspect (what parts the student feel worked well; and what could be improved) was often weak or missing. Also, instructions for how to use the sketch were sometimes missing. Writing these descriptions as requested should have been straightforward, and some students missed out on easy marks by not submitting what was requested here.

One final point to note: many students used assets from other sources in their sketches (e.g. images, sound files, etc.). It is perfectly fine to do this, but it is essential to state where the assets have come from. This can be easily done usually by including comments and URLs within the submitted source code; or in the written description of the code.

Overall, this coursework assignment was generally done to a very high standard, and it was gratifying to see students developing their skills and creativity to produce some very interesting submissions.