Examiners' commentary 2017–2018

CO1112 Creative Computing I: image, sound and motion – Zone B

General remarks

Overall performance on this paper was reasonable to weak, with the average mark being a low second class. A very small number of candidates obtained a first, and there were an alarmingly high number of fails. Close to a third of all candidates did not obtain sufficient marks to obtain a pass. This year, almost all candidates answered exactly four questions, which meant they did not lose marks unnecessarily for leaving out questions or for answering too many.

What follows is a brief discussion of the individual questions on this paper, with hints and explanations of the answers expected by the examiners.

Comments on specific questions

Question 1: General

This was the most popular question, answered by almost all candidates, and with over- all performance of a medium second. A couple of candidates obtained 21 out of the possible 25 marks.

This was a straightforward question, composed of short questions covering a range of topics through the course, and examiners expected a reasonable to good performance on it, demonstrating a general overall competence.

The first part was a simple identification of true or false statements. In the list given, most candidates were able to correctly distinguish that the only false statements were the third, fifth, seventh and eighth. Some candidates incorrectly thought that Andy Warhol was involved in the work at the Bauhaus, despite the fact that his work was in the 1960s, a considerable time after the Bauhaus was developed.

For part (b), disappointingly few candidates were able to identify a woman who had been part of the Bauhaus. Marianne Brandt is the most obvious example, mentioned in the subject guide, but there are many others, including Annie Albers and Florence Henri.

Most candidates were successfully able to perform the required matrix multiplication for part (c), obtaining the answer of:

46 17

18 2

82 37

Some candidates made slight arithmetic errors, and obtained half of the available marks. For part (d), a number of candidates incorrectly included NOT as a bitwise logical operator in *Processing*. Only AND and OR are correct.

Almost all candidates were able to name two different audio compression formats, though some included ones that are principally video or image formats. Where they were video, full marks were still awarded because of the audio component in almost all video. Another common mistake was to give lossy and lossless as the two different formats; they are techniques rather than formats.

Questions (f) and (g) were not answered well. Answers in any format were accepted, but it was essential to show, for part (f), the three bits in one generation as input and the resulting output for the next generation. A good answer might have shown:

111	110	101	100	011	010	001	000
0	0	1	1	1	1	0	0

while for part (g), the output would look like:

```
1
11
1 1
1111
1   1
```

For the final part, answers expected were that the three components are genes (or genetics), development and reproduction, though responses that showed understanding without giving these exact words were also accepted.

Question 2: History and Creative Thinking

The least popular of all the questions, answered by only a third of candidates and also attracting the lowest marks, by quite a large margin. The examiners were disappointed to note that only one of the candidates obtained a pass mark for this question, and the average mark was around 7 out of the possible 25 available.

It is possible that candidates find the requirement of answering discussion-type questions difficult, and avoid attempting or practicing this. You are encouraged to develop an ability to engage with such questions as part of your learning.

For part (a), most candidates knew that circle, square and triangle are the basic forms, and that it is the case that anything can be drawn or constructed from these shapes. Examiners then expected a sensible discussion that connects *Processing* with these ideas, for example that *Processing* provides some basic shapes and we are able to make all kinds of shapes (both 2- and 3-d) from them.

Part (b) required an understanding that visualisation is a graphical representation of knowledge or data. Some candidates incorrectly said that it was illustration, without mentioning the information/data aspect. Good examples include the London tube map (it's a useful representation rather than a geographically accurate map); or a pie chart (it can help with getting an idea of scale of things). Bad examples include the '10% extra free' on a can of coke, that is actually 1/3 of the can, giving an inaccurate impression; pie charts with too many slices to be of any use at all.

One candidate proposed the idea, which was received well by the examiners, that visualisation involves envisaging concepts, and discussed ways of doing this.

Hardly any candidates gave a reasonable answer for part (c), which is actually a very straightforward question that is asked in the subject guide, with relation to John Cage's music. An algorithmic component means using algorithms to choreograph, either explicitly or implicitly. In Cunningham's work, he would give his dancers an algorithm (such as 'two steps left, turn, one step right; repeat 12 times; stop'). There are various ways this could be extended and generalised – such as giving different dancers different algorithms; the algorithms being based on different components (e.g. time, space, music, light) or even being generated during performance, in response to things happening in the theatre or room.

Question 3: Colour

This was chosen by around two thirds of candidates, with an average mark of just above a pass. No candidates performed particularly well, with the highest mark being around 16 out of the possible 25.

Most candidates obtained reasonable marks for part (a) and were able to give examples of Bauhaus members who worked with colour. Itten spoke about colour contrasts and was one of the first to formalise any notion of colour scheme. Paul Klee described a six-part rainbow, and contributed work on colour mixing. Other correct answers were accepted.

For part (b), the examiners expected candidates to understand that RGB is additive, while CMYK is subtractive, and that CMYK is more appropriate for printers, while RGB is more appropriate for monitors. For both, there is a restricted colour gamut but the actual colours in the range differ. Some candidates argued, incorrectly, that CMYK involves four primary colours while RGB involves only three.

Part (c) is material directly from the subject guide, with the different ways including: hash notation; hex with explicit Alpha channel; using the `color()` construction; and the bit- wise operations. An additional mark was awarded for each when the colour green was correctly coded; most candidates were able to include this aspect.

For part (d), secondary colours are the colours that are obtained by mixing pairs of the primary colours. A simple explanation of how they are constructed is by performing an OR of the two primary colours being mixed, though other correct answers were accepted too.

Question 4: Motion and Interaction

This was the second most popular question, answered by most candidates, and with an average of a low second, similar to the overall examination average.

Most candidates were able to draw a reasonable sketch for part (a), including the required items of a green background, and a blue horizontal bar with top-left corner at (0,230) and bottom-right corner at (499,270). The centre of the hole in the bar is at (250,250) and the ball is a red circle with its centre initially at y=0 and x chosen at random from the range (0,499). The score is white text, showing the number 0 at coordinate (250,40). Answers that did not contain all of these aspects lost part marks.

For part (b), many candidates were able to explain that `keyPressed()` is a callback method that *Processing* calls whenever a key is pressed. In contrast, the `keyPressed` inbuilt variable is a Boolean variable that indicates whether any key is currently be- ing pressed; it would normally be used within the `draw()` method – in which case it would be called once per frame. (Unfortunately, many candidates argued incorrectly that `keyPressed()` should be used outside the `draw()` method.) In both cases, you would need to use the inbuilt variable `key` (and maybe also `keyCode`) to determine whether a specific key has been pressed. Marks were given in proportion to how much of the above was covered in the answer, and also for any other relevant information.

A simple answer for part (c), which unfortunately many candidates were not able to give, would be to insert the following code within the `draw()` method:

```
if (ballY > 220 && ballY < 280 &&
      (ballX < holeX-10 || ballX > holeX+10))
{
   resetPositions();
}
```

The main omission was either to not include all four tests, or to include incorrect tests. The test could be inserted at any point within the `draw()` method. Of course, more complex but correct answers were also accepted.

Similarly for part (d), a simple answer would be to insert the following code in the `draw()` method (best at the end of the method, although any reasonable position within the method was acceptable):

```
if (ballY >= 500)
{
   score++;
   resetPositions();
}
```

In this question, many candidates were able to provide this correctly.

For the final part, examiners required candidates to understand that there are two aspects needed. First, you would need to create a global `PFont` object (for example, with the code "`PFont f;`" before the `setup()` method), and initialise it in `setup()` using the `loadFont` method, for example, "`f = loadFont("NewFont-32.vlw");`". After this, also in the `setup()` method, you would need to specify that this font should be used, with the `textFont` method, for example "`textFont(f);`". Many candidates omitted this second step.

Question 5: Generative systems

Another reasonably popular question, attracting the highest marks of all questions attempted on average, this being an upper second.

Part (a) was answered reasonably (trees, plants, and so on.); but many candidates were weak on what features of L-systems make them suitable for the modelling.

Part (b) generally was answered correctly. The three iterations would be:

```
bb
acac
bbbabbba
```

The six missing items for part (c)(i) were usually given correctly, with the exception of the explanation of what `translate(20,0);` is doing. The transformations themselves were usually correctly given. In order, the six required answers were:

```
line(0,0,20,0);translate(20,0);
Move forward length 20 without drawing a line rotate(PI/4);
rotate(-PI/4); pushMatrix(); popMatrix();
```

For part (c)(ii), a drawing similar to the below was expected, with correctly annotated lengths (all length 20) and angles.

The final part of this question attracted reasonable answers, but also some weaker ones. For part (i), examiners expected candidates to know that the population includes a variety of solutions, and that some form of selection is used (either user-guided/aesthetic or a fitness function) to choose the best ones to form the basis of the next generation.

For part (d)(ii), some candidates knew that variety is introduced through mutation, and that mutation alters the values of the genes with a low probability. Candidates were also expected to understand that variety is required as the raw material for selection and evolution.

Part (d)(iii) was sometimes weakly answered. Some candidates repeated stock information from one of the coursework assignments done earlier, and others created a completely new string with no mutation. Examiners expected that the method would make a small number of changes (usually one, but sometimes more). Mutations could include swap- ping a symbol in the given rule string for a different symbol, deleting a symbol from any position, or inserting a new symbol (at the beginning, end, or any position within the existing string). The main problems to look out for are preventing the method from returning an empty string, and ensuring that every [symbol is followed by a matching] symbol.

Question 6: 3D drawing, animation and object-oriented programming

Question 6 was another unpopular question, answered by around a third of candidates, and also attracting very low marks (on average, around 8 out of the available 25).

Part (a) started off well, with most candidates able to say that the required code would be:

```
ax = 0.0; ay = 0.2; az = 0.0;
```

Part (b) was also answered well, with the replacement suggested being:

```
vx += ax; vy += ay; vz += az;
```

Again, most candidates were able to say that of the nine arguments, the first three are the x, y and z position of the camera; the second three are x, y and z of the centre of screen (or focus point), and the final three specify which axis is facing up.

For part (d), again many candidates provided an appropriate drawing, including most of the required annotation. The sketch draws a green ground plane (or a box) of 600 units square (and height 1 unit), with a white background. The camera is located at position `(0,-1000,1000)` and pointing at the origin `(0,0,0)`. A single particle, represented by a blue sphere of diameter 10, starts at the origin and moves upwards in a random direction (that is, with a y velocity between -10 and -20, and x and z velocities between -10 and +10). Its upwards velocity is gradually diminished by gravity and it eventually moves downwards with increasing velocity (there is no collision detection with ground).

It was the final two parts, (e) and (f), where most candidates did not provide insightful answers. For part (e), a simple solution would be to just add two more `Particle` global variables at the start of the code, initialise them in the `setup()` method in the same way as `p1`, and call their `.draw()` and `.update()` methods in the `draw()` method, the same as for `p1`. A more complete answer would use an `array` or an `ArrayList` or similar data structure.

For part (f), a very naive answer would assume that we just need to check if `py > 0.0`, and if so, reverse the sign of `vy`:

```
   if (py > 0.0)
   {
      vy = -vy;
   }
```

which some candidates were able to provide. However, we also need to ensure that `py` returns to a negative position immediately after the collision and that `vy` is only inverted if it is positive, to prevent jitter. So a full answer would be something like:

```
if (py > 0.0)
{
   py = -1.0;
   if (vy > 0.0)
   {
      vy = -vy;
   }
}
```