Coursework commentaries 2016–2017

CO1112 Creative Computing I: image, sound and motion - Coursework assignment 1

General remarks

The coursework assignments this year followed a theme, examining concepts relative to the work of Creative Computing I in the context of the artistic work of Carmen Herrera. The first coursework focused on shape, form and colour, while the second focused on 3-D concepts and fractals. In both, some connection with the history and approach evident in the work of Herrera was required, and many students did this well.

One important purpose of the coursework assignments for Creative Computing I is to give students an opportunity to develop their coding skills in the production of a creative artefact. Another important purpose is to learn how to read diverse material, understand it and synthesise written work from it in an academically appropriate way. Furthermore, this is an opportunity for students to learn more about an area related to the main subject material of this course, and to learn how to make connections between topics.

The Examiners were pleased to see that some students had read widely and had given the topic a great deal of thought. Other students read less widely, but through thought and creativity managed to complete high quality work. The general standard of work submitted was very high, with over a third of students obtaining a mark over 70 per cent. There were some weaker submissions too, with around nine per cent of students failing this assignment, but this a much lower number than in previous years. All of those who failed had omitted at least one part of the coursework; it is in general a good idea to attempt all parts and submit what you've done.

Sadly, a further five per cent were found to have plagiarised, despite the advice provided on the virtual learning environment (VLE) under Study Support.

Comments on specific questions

Part 1

The first part required students to undertake some academic reading and to write an appropriate essay about the work of Carmen Herrera. In particular, there was a requirement to link Herrera's work to the work of the Bauhaus; not all students achieved this, although many did a good job.

Weaker submissions did not take an academic approach and used only websites and popular articles as sources. Wider reading will provide more ideas to build on, and citation and referencing are essential to support your essay.

There were some very interesting submissions, showing both insight and imagination; for example, explicit comments about how Herrera and the Bauhaus differed, rather than only describing ways in which they were similar.

The next three parts required the development of software using *Processing*.

Part 2

Part 2 was intended to be quite straightforward: students had to recreate an image of a specific Herrera work. Many did this well, but many also failed to achieve the correct edges where the black and white colours meet, and some also showed bleeding between the two. A few students who did not manage to create the sharp edges noticed this and commented on it, which was a good thing to do as it showed some self-reflection. Do not simply hope that the Examiners will not notice something that you have not done correctly: we are looking for a demonstration of understanding both of the work and your own ability to achieve what has been asked for.

Part 3

Again, part 3 was intended to be relatively straightforward. It required choosing a colour work of Herrera's from a list and exploring concepts of colour with this sketch as a basis. Surprisingly few students stated explicitly which work they had chosen and left it to the Examiners to figure that out. In general, this was not problematic given the limited range of artworks to choose from, but it is always a good idea to communicate with the Examiners, and this is a good place to begin that practice. Put yourself into the position of someone who might be marking your work, and think what information might be useful for them. (Also, think what information might be tedious for them!)

Part 4

Part 4 was more challenging and required students to develop their work; some very imaginative sketches were submitted. Many students continued the colour theme. Some (the more interesting ones) took the work to a new level. In general, using algorithmic approaches can result in a sketch that can be easily modified, extended or elaborated.

Students were given detailed individual feedback on their submissions and are now encouraged to take the positive work forward and consider developing it further for inclusion in a portfolio of their Creative Computing achievement.

Coursework commentaries 2016–2017

CO1112 Creative Computing I: image, sound and motion - Coursework assignment 2

General remarks

This coursework assignment explored the use of 3D drawing, colour theory, fractals and genetic algorithms, for the purpose of the production of visual art in the style of Carmen Herrera.

The overall performance on this assignment was more mixed compared to coursework assignment 1. Over a quarter of students obtained a mark of over 70 per cent. Around 12 per cent failed this coursework, generally because they had not completed every part of the assignment. It is usually a good idea to attempt all parts and submit what you have done, even if your work is not complete, to show what you have attempted. Sadly, a further three per cent were found to have plagiarised, despite the advice provided on the virtual learning environment (VLE) under Study Support.

Although some students submitted very neat, well-designed and well-commented code, many of the submissions showed plenty of room for improvement in coding standards. In addition to a lack of useful comments, there were many examples of code with inconsistent indenting and spacing. People new to programming may regard the neatness of their code as unimportant, but as you get more experienced with writing code – and, in particular, with **reading** code (possibly written by other people) – you realise that having consistent standards of code presentation greatly helps the reader to understand how the code is structured. In *Processing* in particular there is little excuse for producing poorly-formatted code because there is an *Auto Format* function in the *Edit* menu, which will tidy up your code for you!

Comments on specific questions

Part 1

The first part of the coursework was intended to be straightforward. Examiners expected all six of the required 3D points to be selected at random (within sensible limits along each of the axes). Most students did exactly this, which was perfectly acceptable. However, a few students questioned whether the two rectangles needed to be planar, which would have required more constraints to be imposed upon the positioning of the six points. In retrospect, the question could have been stated more clearly, but the issue was resolved by students asking questions on the VLE course discussion board. This is a valuable resource – please use it to discuss issues that arise during your studies, including if you are in doubt about anything asked in the coursework.

While most students dealt very well with the position of the points, a few placed overly restrictive constraints on their positions, which reduced the variety of outputs achievable. For example, some only allowed the points to vary along the y axis, but kept a constant x position for each point each time the sketch was run. This meant that the variety of shapes produced

was fairly predictable and not as interesting as it might have been.

Most students used the *beginShape / vertex / endShape* approach to drawing the rectangles, as presented in the subject guide. A few used the *rect* command to draw a 2D rectangle, together with one or more 3D rotations to distort the rectangle's appearance. Both approaches were perfectly acceptable. Those who used the first approach sometimes found that strange results were produced by the rendering algorithm processing the vertices in an unexpected order (e.g. producing two small triangles rather than a rectangle). Although the Examiners didn't expect students to prevent this from happening, they expected at least to see some comments about this when it occurred.

A surprising number of submissions used a completely random method to select the colour of each rectangle and the image background. Examiners were expecting to see colour schemes driven by colour theory here; for example, the colour of one rectangle might have been selected at random, and then the other two colours selected relative to the first one by using a three-colour scheme as discussed in the subject guide, Volume 2, Chapter 1.

In the final part of the question, which asked students to select three of their favourite generated images and briefly discuss why they chose each one, some students gave trivial answers such as that they liked how it looked. While Examiners were not expecting a long essay here, they expected some explanation of **why** the student liked the image (for example, the shapes and/or colours could be reminiscent of some natural scene).

Part 2

The second part of the coursework should have been relatively straightforward. Many students did very well on it, but others had problems.

For choosing the colour scheme based upon colour theory, the same problems apply here as stated above for Part 1.

For choosing a random value for the *L* variable between the limits of 0 and 3, many students submitted code that did not work exactly as it should have done. Many students used code similar to the following:

```
int L = int(random(0,3));
```

There is a subtle problem here: the *random()* method deals with floats, and will return a float value between (and including) the value of the first argument and up to (but **not** including) the value of the second argument. Hence, in this case, *random()* will return a value between 0.0 and 2.9999 inclusive. The *int()* conversion always rounds its argument down (towards zero), so even the expression *int(2.99999)* will return 2. Hence, the line shown above will only ever produce the values 0, 1 and 2. A simple solution would have been to use the following line instead:

```
int L = int(random(4));
```

Some other students used the *round()* method instead of the *int()* conversion here, but this too introduced some subtle problems. For example, some submissions used code similar to the following:

```
int L = round(random(0,3));
```

Unlike *int()*, *round()* converts its float argument to the closest integer (so it will round up or down as appropriate). For example, *round(2.4)* produces *2*, and *round(2.6)* produces 3. So, while the code shown above

will produce numbers between 0 and 3, it will not do so with uniform probability. Consider that only values between 0.0 and 0.5 will produce a result of 0, and only values above 2.5 will produce a result of 3, whereas values between 0.5 and 1.5 will produce a result of 1, and those between 1.5 and 2.5 a result of 2. Hence, the line shown above will only produce results with values of 0 and 3 with 1/6th probability each, where 1 and 2 will be produced with 2/6th probability each.

These subtle problems highlight the importance of proper testing of your code to ensure that it is behaving as expected.

Most students coped well with the second set of questions about adding interaction to the sketch. A few students used the *mousePressed()* function incorrectly: while this is indeed the appropriate place to handle mouse interactions, note that this is a *callback function* that Processing will run automatically at each frame of the sketch. Hence, you just need to implement the function, but you do **not** need to (and indeed **should not**) explicitly call it from within your own code, which some students did.

A disappointingly small number of students made a serious attempt at the final part of this question about further exploration of the code (e.g. by looking at different fractals). This should have been fairly straightforward.

Part 3a

The general standard in this part of the coursework assignment was disappointing. Some students discussed the general design of genetic algorithms but did not relate it specifically to the code in the coursework. Others did go into more specific details and some answered the question well. However, many of the answers revealed some misunderstandings about genetic algorithms in general. A few students suggested implementing a recursive developmental stage using L-Systems, to get from the genetically-encoded parameter values to the final image. While this would have made sense for applying genetic algorithms to the Part 2 code (indeed, that already used an L-System), it did not make much sense in relation to code from Part 1, where the generation of the image from the specified points was much more straightforward. The Blind Watchmaker algorithm described in the subject guide Volume 2 – which uses a recursive developmental stage – is just one example of a genetic algorithm, but there is no general requirement for genetic algorithms to include such a recursive mapping from ‘genotype’ to ‘phenotype’.

Part 3b

A few students wrote excellent answers to this question, showing evidence of extensive research into Herrera’s painting process, and thoughtful insight into the similarities and differences between her work and images that have been produced algorithmically. Examiners expected appropriate use of citation and referencing here, to indicate any sources of information used in the answer – some students did a much better job of this than others. There were many interesting points of comparison between Herrera’s work and algorithmically generated images. To give just one example, Herrera’s process usually involves generating a large number of images, and then very selectively choosing the ones she likes for further work. Many students raised the point that part of the value of art comes from the intention of the artist who generated it – this is an important point, and yet many submissions only mentioned it superficially, without any further exploration.

Overall, this coursework was generally done to a high standard, and it was gratifying to see students developing their skills and creativity to produce some very interesting submissions. It appeared that most of the students who did not perform so well had simply not allocated sufficient time to complete the work properly.