# Examiners' commentary 2018–2019

## CO3355 Advanced graphics and animation – Zone B

### General remarks

This year, the general standard of performance was lower compared to last year, with about 2 out of 3 candidates achieving a pass mark. However, the percentage of excellent attempts was significantly higher.

In terms of individual popularity, Question 1 was the most popular with 88 percent of the candidates attempting it, followed by Question 5 (73 percent) and Question 2 (62 percent). About 42 percent of the candidates answered Question 4, while Question 3 was the least popular, with around 30 percent of the candidates attempting it.

In terms of average performance, Questions 3 and 1 attracted the best answers, followed by Questions 2, 5 and 4 (in that order).

As in previous years, most candidates showed appreciation of the relevant issues in questions but many failed to provide clear and concise explanations. Practising such clarity in explanations is an area worth focusing on in revision and examination practise, and it should build upon the skills developed in earlier assignments and examinations.

### Comments on specific questions

#### Question 1: Maths and transformations

a. Part (a) tested fundamental knowledge of vector mathematics, and the majority of candidates answered correctly.

b. Here, candidates had to transform a vector by performing a clockwise rotation by π radians around the origin. A surprisingly common mistake here was to consider π radians to be 90 degrees.

c. In part (c) a pair of vectors was given, and candidates were asked to calculate their distance and angle. Many candidates had difficulties here, especially in the first part (i).

d. The statement in part (d) was false. Homogeneous coordinates allow representation of operations as matrix multiplications. However, this does not mean that these operations are commutative. In other words, the order of application does matter. A verbal explanation of this would suffice. Another way to disprove the statement was to multiply two homogeneous matrices showing that commutativity does not hold.

e. This part examined understanding of basic geometric transformations. Candidates had to identify the components of a 2D matrix operation, making sure they respected the order of their application, that is from right to left. It sufficed to say that first the vector was translated by [-2 -2], then rotated by 90 degrees and, finally, translated by [2 2]. The effect of the combined operation was a rotation by 90 degrees about a fixed point ([2, 2]).

   While most candidates identified the transforms correctly, sometimes the order of their application was either incorrect or not clear. With respect to the purpose of the combined operation, very few candidates managed to provide a clear explanation. Some identified the similarity with a transformation stack and those who provided a clear justification for this earned full marks.

## Question 2: Model to screen

a. This was a bookwork question and was mostly well answered.

b. Part (b) asked candidates to consider why we would consider using orthographic projection. Pointing out that parallel lines are kept parallel on screen and mentioning applications such as engineering and architecture would yield full marks.

c. The operation described in part (c) was culling. It is important as it reduces the number of rendered polygons and, thus, computational complexity.

d. Part (d) examined hidden surface rendering techniques.

   i. A good answer would explain the theory (bookwork – see subject guide p.35) with simple reference to the surfaces shown in Figure 1. However, the majority of candidates failed to provide a concise and clear explanation of this as many thought that the surface representation in the figure was problematic and had to be solved by the depth buffering algorithm.

   ii. A concise explanation of the z-buffer algorithm, using surfaces *A*, *B* and *C* would attract full marks. A common mistake here was the impression that z-buffering considers the depth of each surface or object (and not each pixel).

   iii. The image would look the same, regardless of the order in which the primitives are rendered, as z-buffer solves the visibility problem. Despite some good responses here, the majority of candidates did not manage to correctly justify their answer.

e. The final part of this question tested candidate's generic knowledge on established 3-dimensional test models. Any justified answer could work here, as long as it was supported and well explained. Characteristics might include: number of polygons (large to test detail or precision and/or speed or low for quick and easy testing of material, animation, textures or lighting, and so on), convex or concave curvature, manifold joints, imperfections to make it more realistic, multiple surfaces suitable for testing indirect/global illumination, and so on. Unfortunately, many candidates discussed generic testing processes instead of model objects and were awarded partial marks. Despite that, there were also some excellent answers.

## Question 3: Graphics programming

a. Although a few candidates skipped this part of the question, it was generally well answered.

b. In part (b) candidates had to name three types of operations/effects fragment shaders are suitable for, and most did so in a competent manner.

c. Part (c) examined three types of shapes, i.e., triangles, triangle strips and triangle fans. Although most candidates seemed to understand their differences, many had problems, particularly in (ii), while others did not realise that the origin of the screen coordinate system is at the top left corner.

d. This tested basic GLSL programming.

   i. A good answer here would state that the code loads a shader from file "fshader.glsl" creates a sphere, positioned in the centre of the frame and applies the shader on it.

   ii. Here a very simple fragment shader needed to be specified, such as the following example:

```
void main() {
    gl_FragColor = vec4(1.0,0.0,0.0,1.0);
}
```

iii. An answer to this one could be using the following shader:

```
uniform float t;
void main() {
    gl_FragColor = vec4(abs(sin(t)),0.0,0.0,1.0);
}
```

and setting the variable inside the Processing program:

```
sh.set("t", millis() / 1000.0);
```

This was mostly well answered. Most common mistakes were not making sure that the blue colour component was a value between 0 and 1.

## Question 4: Lighting and display

a. This bookwork question was mostly answered well. See p.90 of the subject guide.

b. Part (b) asked candidates to compare and contrast image-based lighting and environment mapping, and attracted mostly good answers. See p.81 and p.90 of the subject guide.

c. In part (c), candidates were asked to provide a brief definition of *flat shading*. This was mostly well-answered. See pp.67–68 of the subject guide.

d. Here, candidates had to craft an ideal situation where flat shading would be accurate. Any justified and well explained example could gain full marks. For example, flat shading could be accurate if there was infinite computing power, and could use 'infinite' resolution for the model. Another example could be having the light source at infinity, so that the light vector was constant, while the viewer was at infinity and polygons represented actual surface, not an approximation. Although many candidates described situations where it would be somewhat adequate (such as when low complexity is a must), only a few did indeed craft situations where it would yield full representational accuracy.

e. In part (e) candidates were presented with Figure 2 – Triangle shading.

   i. Candidates were asked to describe the steps involved in calculating the colour of a pixel using Gouraud shading. A good answer would expose the following steps:

      - Calculation of normals and colour at points *A*, *B* and *C*.
      - Performing linear interpolation to find the colour at proper points along the edges *AB* and *AC*.
      - Performing linear interpolation along the horizontal scanline to find the colour of pixel *P*.

   A diagram with scanlines and points should also be included, while colour interpolation equations would help compensate for any lost marks. This was generally answered well.

   ii. Part (ii) asked what problem emerges if light from a narrow directional source hit *P* directly, but dies not hit *A*, *B* or *C*. If a light source hits an edge or a face of a triangle, but does not hit the vertices, it will not be visible at all. This can be resolved by using per-pixel shading. Very few candidates identified the problem, but those who did proposed a valid solution.

## Question 5: Texturing

a. Part (a) asked for the difference between texture mapping and procedural texturing and was generally answered well. See pp.74–79 of the subject guide.

b. Part (b) focused on texture mapping.

   i. This part asked candidates to identify and briefly describe any three coordinate systems, as long as they are somehow involved in texture mapping. Although there were some very good answers, in most of them the connection with texture mapping was not clear.

   ii. Here, candidates were asked to describe the roles of fragment and vertex shaders in texture mapping. Good answers identified that the vertex shader must transform the texture coordinates of the vertex, while the texturing itself is done in the fragment shader which uses the transformed texture coordinates to sample the texture and (possibly) combine it with lighting. Many candidates described the role of each shader type but did not relate it with texture mapping.

   iii. This part was about the *Sampler2D* variable, which is used in the fragment shader as a uniform variable and samples from the texture. About half of the candidates provided good answers here.

c. This was a bookwork question part about texture mapping with a cylindrical map shape, and was mostly well answered. See subject guide, p.75.