Examiners' commentary 2017–2018

CO3355 Advanced graphics and animation – Zone B

General remarks

This year, the general standard of performance on this examination paper was on a par with the previous one. About eight out of ten candidates achieved a passing mark, including a number of very strong papers.

In terms of individual popularity, Question 1 was the most popular with 90 percent of the candidates attempting it, closely followed by Question 2 (80 percent) and Question 4 (77 percent). About 40 percent of the candidates answered Question 5, while Question 3 was clearly the least popular, with only around 13 percent of the candidates attempting it.

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

Comments on specific questions

Question 1

Maths and transformations

a. This tested fundamental knowledge of related mathematics and the majority of candidates answered well, though some had issues with vector subtraction.

b. This was a bookwork question about homogeneous coordinates, which are used extensively in computer graphics because they allow common operations such as translation, rotation, scaling and perspective projection to be implemented as matrix operations.

   Most found the concept difficult to explain, including some that confused it with uniform/non-uniform scaling.

c. This part examined the understanding of basic geometric transformations.

   i. This matrix represented a translation by [3, 1, 3]. The result of the calculation would be $[4, 2, 5, 1]^T$.

   ii. This represented a scaling by [2, 2, 3] and the result was $[2, 2, 6]^T$ or $[2, 2, 6, 1]^T$.

   iii. This represented a 60 degree rotation around the z axis, followed by translation by [2 0 0]. The resulting vector was $[3.37, -.37, 2,1]^T$.

   The vast majority provided very good answers and achieved full marks. Unfortunately, some candidates appeared to have issues with basic algebraic concepts and especially matrix multiplication.

d. This part examined generic 3D transformations and the effects of their order of application.

   i. Here a single, generic matrix that combined the two transformations was asked for.

   Although there were many good answers, some candidates produced two separate matrices (i.e. one per transform), while others applied the matrix on a vector and calculated the transformation result. That was not necessary.

ii. For transforms of this type, combining rotation followed by a translation along the y-axis, the result is not affected by the order of their application.

iii. In order to prove the previous claim, it sufficed to show that this type of transform is *commutative*. In other words, assuming two such transforms G_1 and G_2, one had to show that $G_1*G_2==G_2*G_1$.

Some candidates, instead of using combined transforms to prove the previous statement, examined whether changing the order of performing rotation (R) and translation (T) affects the result. Thus, they checked whether R*T is equal to T*R. Those who did so correctly, were awarded most marks.

There were also a number of candidates who worked on combining scaling and translation transforms. However, scaling was not relevant to this question.

Question 2

Model to screen

a. This part examined *Object* and *Camera coordinates*, asking candidates to specify how a model representation can be converted from the former to the latter.

While most explanations of coordinate systems were competent, many candidates failed to identify the ModelView matrix (or, equivalently, a combination of Model and View matrices) as the means for conversion.

b. This part asked for two example scenes where the use of local coordinate systems is helpful. Any valid example would be appropriate here (modelling hierarchical objects, aiming to support independent transformations, and so on.) Only a few answers gained full marks, as a large number of them only presented one single example scene.

c. This part examined candidates' understanding of how the inherently discrete computer representation affects drawing of curves. The statement presented is generally true, because of rasterisation, i.e. the process of converting vector representations to discrete ones. The quality of the representation can vary greatly, depending on the resolution.

Most candidates answered correctly but some lost marks because they did not illustrate their answer with an example.

d. This part examined the process of *culling* and how it differs from *clipping*

Most candidates answered well. However, some did not identify that this is done by Processing automatically.

e. The order of drawing objects does indeed affect the efficiency of the z-buffer; ideally, objects should be drawn from front to back. This reduces computations, avoiding frequent modifications in the colour and depth values of the algorithm, as the depth test would normally fail. Only a few candidates provided clear answers here.

f. A good answer here would specify and briefly describe the four points needed to define a cubic Bézier curve. An illustration could also help to gain lost marks. This was mostly well answered.

Question 3

Graphics Programming

a. Only two commands were needed here, namely:

```
fill(255,0,0)
sphere(100);
```

The majority answered well.

b. This discussed how *graphics objects* can tackle some issues of *immediate mode* and was generally answered well.

c. This examined the term graphics state in Processing. A good answer would explain how certain commands do not affect particular shapes but rather change the state of the renderer, thus affecting any shapes drawn after that, and include two examples of such commands.

   This was mostly well answered.

d. This examined how programmable shaders can facilitate certain effects compared to traditional, fixed GPU functionality, asking for three examples. A few of them could be found in the subject guide, such as complex materials, natural phenomena, advanced lighting, non-photorealistic rendering, and animation.

e. This tested awareness of existence of other shader languages. One could mention HLSL by Microsoft (used in DirectX), Cq by Nvidia, PSSL by Sony (used in PlayStation), OpenGL ARB, and so on.

f. In this part, the code of a shader was provided. Candidates had to identify that it was a *fragment* shader that changes the current colour to Yellow. In order to change the opacity, one could modify the last parameter of the corresponding vec4 variable.

   With respect to (iii), an answer along the lines of the following would be appropriate:

```
PShader shader = loadShader("shader.glsl");

shader.set("mousePos", 100*float(mouseX)/
float(width),  100*float(mouseY)/float(height));
```

   Finally, one could manage progressive modification of the colour from Red to Green, according to the mouse position with a command similar to the following:

```
color = vec4(mousePos.y/100.0, (1.0 -
mousePos.y/100.0), 0.0, 1.0);
```

   (i) and (ii) were answered well by almost all candidates. On the other hand, many faced problems with scaling in (iii) where changing the display size to (100,100) was not an option. Finally (iv) was answered only by a small percentage of candidates.

Question 4

Lighting and display

a. This examined the term *specular reflection* and was generally answered well.

b. This explored the differences between global and local illumination and was mostly well answered.

c. BRDFs can be measured directly from real objects using specially calibrated cameras and light sources, or they can be models derived from empirical measurements of real-world surfaces. Mentioning one of the two methods would be sufficient here.

   Only a few answered this correctly. Some described what a BRDF was but this was not the focus of the question.

d. This part examined advanced illumination, asking candidates to select between *radiosity* and *ray tracing* to illuminate a specific scene.

   As ray tracing is entirely dependent on the viewer's position, it involves recalculating the lighting by processing the entire scene every time the

viewpoint changes. On the other hand, radiosity calculates indirect lighting independently from the viewpoint, so it would be much less computationally expensive, making it the best choice in this case. This does not depend on the indoor/outdoor setting, as we disregard the quality of the result.

Most identified that radiosity would be more adequate, being less computationally expensive. However, many answers lacked clarity, particularly failing to associate the complexity with the fact that the camera is constantly moving.

e. This part asked candidates to devise an example of a scene where Phong shading and Gouraud shading would produce different results. A good example would consider a situation with high amounts of specular reflection, as Phong handles it better than Gouraud, for instance the case of a perfect mirror face and a point light source. Identifying (i) specular surfaces and (ii) polygon count were the key factors here.

This was mostly answered well. Some described how calculations are made in each method but that was not required.

Question 5

Texturing

a. This part asked about the term *image map* and was generally answered well.

b. Here candidates had to name the two main methods to surface texturing (i.e. *texture mapping* and *procedural texturing*).

c. This part examined the *heightfield* method in the context of *bump mapping*. Very few candidates provided clear answers.

d. This examined the term *map shape* and was generally answered well.

e. Candidates here had to identify that the mapping equations corresponded to a planar map shape, where the y-axis was ignored. Moving along that axis would not affect the colour of the object.

This was mostly answered well.

f. This provided an example scene and asked to identify the factors one would take into account when faced with a choice between environmental mapping and ray tracing. A good answer would explain how each method affects complexity and representational quality.
Most candidates answered well.

g. This part discussed *environment mapping*. Candidates had to assess whether it constitutes an approximation to true reflections. A good answer would explain how it simulates reflections by using a texture, based on the direction of the reflected view vector. The result is approximate reflections from distant objects. However, reflections in nearby objects or the object itself are not captured. Furthermore, the environment map is only correct from one unique point of the scene, i.e. the point the cube map was generated from. Commonly, the cube map is static, or at least not updated for each frame. Finally, the spherical reflection map is often represented with a cube-map, where the sphere is approximated with six faces.

Due to all these reasons, environment mapping is indeed just an approximation of true reflections.