Examiners' commentary 2017–2018

CO3355 Advanced graphics and animation – Zone A

General remarks

This year, the general standard of performance on this examination paper had improved. About seven out of ten candidates achieved a passing mark, including a few exceptional attempts.

In terms of individual popularity, Questions 1 and 4 were the most popular, with about 88 and 76 per cent of the candidates attempting them respectively, followed by Question 2 (about 58 per cent). Questions 5 and 3 were less popular, attempted by 41 and 35 per cent of the candidates respectively.

In terms of average performance, Questions 2 and 3 attracted the best answers, followed by Questions 1 and 4. Question 5 proved to be more challenging, attracting the lowest marks.

Comments on specific questions

Question 1

Maths and transformations

a. This tested basic knowledge of related mathematics and the majority of candidates answered well.

b. This was a bookwork question regarding 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.

   Some candidates chose to demonstrate their necessity by showing how translation can be calculated with and without their use.

c. This part examined understanding of basic geometric transformations.

   i. This matrix represented a scaling by [3, 2, 1]. The result of the calculation was $[1.5, 4, 1]^T$ or $[1.5, 4, 1, 1]^T$.

   ii. This matrix represented a translation by [1 1 0], and the result was $[1.5, 3, 1, 1]^T$.

   iii. This matrix represented a 45° rotation around the z axis, followed by a translation by [2 0 0]. The resulting vector was $[3.75, 1.05, 1, 1]^T$.

   Although this part was mostly well answered, some candidates claimed that in case (ii) this was an identity matrix that would not modify the position of the vector. That's clearly not the case because of the elements in the last column that denote translation. These elements would have been zero in the case of an identity matrix.

d. This part tested understanding of the effects of the order of application of 3D transformations.

   i. For both cases 1 and 2, concerning translation and scaling respectively, the result is not affected by the application order.

ii. In order to prove that claim, it sufficed to show that both of these transformations are commutative, i.e. that $T_1*T_2==T_2*T_1$ and $S_1*S_2==S_2*S_1$.

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

Others did not use generic transforms, which resulted in the loss of some marks.

Some also modelled the position vector and applied the transforms on it. This was not necessary, as it was the transform itself that was of importance, and that is fully modelled by the matrix and not affected by the vector(s) it is applied on.

Question 2

Model to screen

a. This part examined two fundamental transformation matrices used in Computer Graphics. The *ModelView* matrix is the concatenation of *Model* and *View* matrices. The former defines the position (location and orientation) of the camera, while the latter defines the frame's position of the primitives drawn. ModelView is used to convert Object coordinates into Camera coordinates. On the other hand, the *Projection* matrix defines characteristics of the camera, such as clip planes, field of view, projection method, and so on. and is used to calculate Screen coordinates

   These concepts proved to be somewhat difficult for many candidates.

b. The reason for using two separate matrices, instead of one, is that additional calculations (such as lighting) need to take place in between the two transforms. Only a few managed to answer this correctly

c. This part examined understanding of how the inherently discrete computer representation affects drawing of lines. 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. However, in the case of lines there are two exceptions that can be represented accurately: vertical and horizontal lines. That is because of the rectangular shape of the sampling grid that is used. A diagram demonstrating these would be necessary for a complete answer.

   This part was generally well-answered.

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. This required candidates to name three algorithms aiming at the hidden surface removal problem, such as painter's algorithm, z-buffer, BSP trees, and was answered well by most candidates.

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.

Question 3

Graphics Programming

a. Only two commands were needed here, namely:

```
noFill();
```

```
box(100);
```

Some candidates missed the first command, while others tried cube(100). There were a few who drew the cube vertex-by-vertex, also attracting full marks.

b. This part asked about the properties contained in graphics objects and the Processing class used to represent them.

While most candidates correctly identified PShape as the Processing class, most had difficulties in naming the properties (such as geometry and material and in some cases transformations applied to geometry).

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

Unfortunately, very few candidates answered well.

d. GPUs are designed for massively parallel calculations, as opposed to CPUs that work mostly sequentially. Graphics calculations are fit for parallel processing as they can be performed independently (e.g. per vertex or fragment).

e. 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.

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 Magenta. In order to modify the colour opacity, one could modify the last parameter of the corresponding vec4 variable.

With respect to (iii) an answer in the line 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 Blue to Red, according to the mouse position with a command similar to the following:

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

This question was mostly well answered. Some candidates chose to use the map() function for scaling the mouse position.

Question 4

Lighting and display

a. This examined *ambient illumination* and was answered well by the vast majority of the candidates.

b. This part asked about *diffuse reflection*.

   Although it generally attracted good answers, some did not seem to grasp the difference between diffuse and specular reflection, describing the latter instead of the former.

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. However, not many managed to do so.

d. This part examined advanced illumination, asking candidates to choose 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 to devise an example of a scene where Phong shading and Gouraud shading would produce different results. A good example would expose 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.

Question 5

Texturing

a. A good answer here would explain that when using plain shading it is theoretically possible to achieve similar levels of realism as the ones achieved with texturing, but that would come at a cost of higher polygon counts. On the other hand, the generative approach of procedural texturing can promise infinite resolution.

   Most candidates answered well.

b. This examined the difference between noise() and random() functions in Processing. It sufficed to say that the former generates Perlin noise, while the latter generates a normal one. Most answered well.

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

d. 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.

While most identified the type correctly, some missed the second part of the answer.

a. This part asked about the role of interpolation in texture mapping, requiring an example of such a situation and asking candidates to specify which type of uniform variable is used in GLSL for this purpose.

   Here candidates could mention that sometimes we may calculate (u, v) coordinates that are not directly at the pixels in the texture, but in between, making interpolation necessary. This can occur for instance when the image is magnified or minified with respect to its original resolution. A uniform Sampler2D variable is usually used in the fragment shader for this purpose.

   While many candidates described the general principles of interpolation and its use, this question was focused on its role in texture mapping. Only a few managed to gain full marks.

b. This provided an example scene and asked candidates to identify the factors that 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 students answered well here.