Degree course in Computer Science
University of Calabria, Italy

Written exam of Computer Graphics

(September 8, 2017)

Exam duration: **one hour**
NOTE: **The use of any documentation is prohibited**.

Name:____

Surname:____

Student' ID:____

Signature (required at the bottom of each page,)_________________

____

Notes (optional):_____________________________________

(1) Mark the correct statement:
- (a) OpenGL (Open Graphics Library) is a cross-platform application programming interface (API) for rendering 2D and 3D vector graphics.
- (b) OpenGL (Open Graphics Library) is a cross-platform application programming interface (API) for rendering 2D and 3D raster graphics.
- (c) OpenGL (Open Graphics Library) is a cross-platform application programming interface (API) for rendering 3D vector graphics only.
- (d) OpenGL (Open Graphics Library) is a cross-platform application programming interface (API) for rendering 2D raster graphics only.

(2) An OpenGL core profile graphics application is composed by an host and a device application, this latter written in the OpenGL Shading Language (GLSL). Mark the correct statement:
- (a) The device application is defined by at least a vertex and fragment shader. The vertex shader processes vertex data in parallel, while the fragment shader processes fragments sequentially (because processing the generic fragment *i* requires the fragment *i*-1 has already been processed).
- (b) The device application is defined by at least a vertex and fragment shader. The vertex shader sequentially processes vertex data (because processing the generic vertex *i* requires the vertex *i*-1 has already been processed), while the fragment shader processes fragments in parallel.
- (c) The device application is defined by at least a vertex and fragment shader. The vertex shader processes vertex data, while the fragment shader processes fragments. Both of them are executed in parallel (because processing a given vertex/fragment does not require other vertices/fragments have already been processed).
- (d) The device application is defined by at least a vertex and fragment shader. The vertex shader processes vertex data, while the fragment shader processes fragments. The vertex shader processes the first vertex and then the fragment shader is called to process the resulting fragment. The process is iterated over all the remaining vertices/fragments.

Signature: ________________________________

(3) A Model transformation transforms:
- (a) fragment data from Model Space to Window Space.
- (b) vertex data (e.g. position, normal vector, etc.) from Vector Space to World Space.
- (c) vertex data (e.g. position, normal vector, etc.) from Model Space to World Space.
- (d) vertex data (e.g. position, normal vector, etc.) from Model Space to View Space.

(4) A View transformation transforms:
- (a) fragment data from View Space to Window Space.
- (b) vertex data (e.g. position, normal vector, etc.) from Vector Space to View Space.
- (c) vertex data (e.g. position, normal vector, etc.) from View Space to World Space.
- (d) vertex data (e.g. position, normal vector, etc.) from Model Space to View Space.

(5) GLSL applications essentially process buffers. In particular, Vertex Buffer Objects (VBOs) are buffers used to store vertex data attributes like positions, colors, normals. etc.
- (a) A single VBO must be used to store vertex data in interleaved fashion. Offsets and strides must be specified for each specific vertex attribute to allow the vertex shader to properly access the data.
- (b) Different VBOs must be used to store different vertex data attributes. In other words, attributes in a single VBO are always tightly packed.
- (c) Different tightly packed VBOs or a single interleaved VBO can be equivalently used to store vertex data.
- (d) A single VBO must be used to store a single tightly packed vertex attribute at a time and different GLSL programs must be used to process different VBOs.

(6) The Vertex Shader:
- (a) Processes fragments and patches by applying GLSL built-in transformation algorithms to obtain object space positions for the incoming vertices; fragments and patches are independent to each other and therefore they are generally processed in parallel by many different shader cores.
- (b) Processes objects' complex geometry to breack it into independent vertices ; Geometries for different objects are independent to each other and therefore they are generally processed in parallel by many different shader cores.
- (c) Processes raster fragments of the objects into the graphic scene by converting them into set of vertices in an ideal vectorial representation of the object itself; Fragments belonging to a single object are processed in parallel by different shader cores, while object are elaborated sequentially.
- (d) Processes vertex data, such as position, normal vector, etc., and is mainly used to apply to them model-view and projection transformations; vertices are independent to each other and therefore they are generally processed in parallel by many different shader cores.

(7) After the Rasterization stage:
- (a) Vertices and normals have been transformed into world space coordinates.
- (b) Vertices and normals have been transformed into object space coordinates.
- (c) The geometry has been converted from what is essentially a vector representation into a large number of independent *fragments.*
- (d) A color is assigned to each *fragment.*

Signature: ____________________

(8) Given material with ambient term k_a, diffuse term k_d, specular term $k_{s,}$ and shininess factor α, and a light p with ambient term i_a, diffuse term i_d, and diffuse term i_s, the Blinn-Phong lighting mdel is expressed as:

$$I_p = k_a i_a + k_d i_d \boldsymbol{L} \cdot \boldsymbol{N} + k_s i_s (\boldsymbol{H} \cdot \mathbf{N})^{\alpha}$$

where

- I_p is the resulting color at fragment p
- $\boldsymbol{N}$ is the surface nurmal at fragment's coordinates
- $\boldsymbol{L}$ is the unit vector from the fragment towards the light point
- $\boldsymbol{V}$ is the unit vector exactly halfway between the view direction and the light direction

Please, comment the above equation by listing and briefly describing the physical phenomena it (roughly) models.

(9) The Phong lighting model only differs from the Blinn-Phong one for the evaluation of the specular component. Its equation is the following:

$$I_p = k_a i_a + k_d i_d \boldsymbol{L} \cdot \boldsymbol{N} + k_s i_s (\boldsymbol{R} \cdot \boldsymbol{V})^{\alpha}$$

Please, explain what the R and V vector are and the problem the specular component it can trow with respect to the .

Signature: ______________________________

(10) Why both the Phong and the Blinn-Phong lighting models are not able to generate shadows? Which is one simple shadow algoritm? Outline it below (just the main idea. No code is required).

Signature: ____