Please read these instructions carefully!

- Answer the questions in the separate answer sheet.
- You may use the back of all sheets as drafting area (*rascunho*).
- How to:

	A	B	C	D	E
Select the answer (A):	(●	○	○	○	○)
Replace the answer (A) by (C):	(✗	○	●	○	○)
Cancel (C) and reactivate (A):	(✗	○	✗	○	○)

- This test has 26 QUESTIONS (ignoring question zero), each question has a score of 200/26 points.
- **POINTS LOST** for each wrong answer (in percentage of the question's points):

$\sum wrong$	$1 = 0\%$	$2 = 11.11\%$	$3 = 22.22\%$	$\geq 4 = 33.33\%$.

Name: _____ Number: _____

0. Test version. This is not a question and will be ignored for grading!

   A. B. C. ⇒ ● ⇐ D.

Consider a traditional machine with a Central Processing Unit, Main Memory and a graphics card with a programmable GPU (Graphics Processing Unit) with its own memory and supporting WebGL.

1. Choose the correct ending for the following sentence: "The image that a WebGL application is generating at a certain point is stored..."

   A. in a buffer called framebuffer, stored in GPU memory.
   B. in a buffer called frame buffer, stored in main memory.
   C. in a buffer called depth buffer, stored in main memory.
   D. in a buffer called depth buffer, stored in the GPU memory.

2. Choose the correct order for the stages of the programmable pipeline!

   A. Fragment Processing → Primitive Assembly & Clipping → Rasterisation → Vertex Processing
   B. Vertex Processing → Primitive Assembly & Clipping → Rasterisation → Fragment Processing
   C. Vertex Processing → Rasterisation → Primitive Assembly & Clipping → Fragment Processing
   D. Vertex Processing → Primitive Assembly & Clipping → Fragment Processing → Rasterisation

3. From the stages presented in the previous question, two of them are programmable. Which ones?

   A. Vertex Processing and Primitive Assembly & Clipping
   B. Rasterisation and Fragment Processing
   C. Primitive Assembly & Clipping and Rasterisation
   D. Vertex Processing and Fragment Processing

4. Choose the pipeline stage where transformations between coordinate systems is performed most often!

   A. Rasterisation B. Fragment Processing C. Vertex Processing D. Primitive Assembly & Clipping

5. Choose the pipeline stage where projection takes place!

   A. Vertex Processing B. Fragment Processing C. Rasterisation D. Primitive Assembly & Clipping

6. Choose the pipeline stage where the information passed in the first argument of `drawArrays(mode, ...)` or `drawElements(mode, ...)` is certainly used!

   A. Vertex Processing B. Rasterisation C. Primitive Assembly & Clipping D. Fragment Processing

7. Choose the pipeline stage that is responsible for performing the interpolation of the outputs of a vertex shader!

   A. Rasterisation B. Primitive Assembly & Clipping C. Fragment Processing D. Vertex Processing

8. Choose the pipeline stage where the location of a fragment in the output image is determined!

   A. Frament Processing B. Primitive Assembly & Clipping C. Rasterisation D. Vertex Processing

9. Choose the description that best describes the inputs to a fragment shader!

   A. The outputs provided by the vertex processing stage after interpolation throughout the primitive.
   B. The values that the application sends to the GPU as attributes.

C. The indices of the vertices that are used for the processing of the current primitive.

D. The values that the application sends as uniform values.

10. Choose the most appropriate description for the possible contents of a single buffer that the application passes to the GPU!

A. The values of all the attributes for all the vertices involved in a specific drawing call and the indices for all the vertices involved in that drawing call.

B. The values for all the uniforms that need to be passed to the GLSL program.

C. The values of a single attribute for all the vertices involved in a specific drawing call.

D. The values of one or more attributes for all the vertices involved in a specific drawing call or, alternatively, the indices of the vertices involved in a specific drawing call.

---

11. Consider the GLSL (Graphics Library Shading Language) piece of code presented on the right. Local variables 'c' and 'p' have previously been declared with the correct type to make the presented code valid. What is the value that gets assigned to 'p'?

```
c = 2.0 * vec3(0.25, 2.0, 1.0);
c.yz = c.xy;
p = c.bgr;
```

A. (0.5, 4.0, 2.0) B. (4.0, 0.5, 0.5) C. (2.0, 4.0, 0.5) D. (0.5, 0.5, 4.0)

12. Consider an application that needs to draw thousands of convex polygons (not necessarily triangles) in the most efficient manner. Ignore the cost of any preprocessing that may be required. How would you handle the problem?

A. Convert each non triangular polygon into a set of triangles to be drawn with a `gl.TRIANGLE_STRIP` primitive.

B. Convert each non triangular polygon into a set of triangles to be drawn with `gl.TRIANGLES` primitives.

C. Convert each non triangular polygon into a set of triangles to be drawn with a `gl.TRIANGLE_FAN` primitive.

D. Use the data without any conversion and draw each polygon with the `gl.POLYGON` primitive.

13. The technique known as *double buffering* is used to support animation. Choose the correct option!

A. None of the other options is correct.

B. It uses two buffers, one where the primitives are drawn and another to be used as a depth buffer to solve the visible surfaces problem.

C. It draws on both framebuffers, one corresponding to the left eye view and the other to the right eye view, thus supporting double/stereoscopic vision.

D. It uses two framebuffers and, while the primitives are being drawn in one of them, the other is being displayed. At the end of a frame, the roles of the two framebuffers are switched.

14. A WebGL application will be used to simulate a **single** rocket in 2D that is launched at point $\mathbf{p} = (x_0, y_0)$ at instant t_0, with initial velocity $\mathbf{v}_0$, exploding in the air at instant t_1. Each piece k will continue with a new initial velocity given by $\mathbf{v}_1^{(k)}$ after the explosion. The simulation is performed by displaying a fixed number of points (thousands) with the `gl.POINTS` primitive and using only one GLSL program. Each point will represent a small piece of the rocket and its trajectory is governed (before and after the explosion) by the set of equations on the right. Note that even before the explosion, we can display the rocket by using its thousands of smaller pieces as long as they move together. Choose the option that best describes the type of variables to be used.

$$x = x_i + v_{i_x}$$
$$y = y_i + v_{i_y} t - \frac{g}{2} t^2$$

A. $uniforms = \{\mathbf{p}, t_0, t_1, \mathbf{v}_0\}$, $attributes = \{\mathbf{v}_1\}$.

B. $uniforms = \{\mathbf{p}, t_0, \mathbf{v}_0\}$, $attributes = \{t_1, \mathbf{v}_1\}$.

C. $uniforms = \{p\}$, $attributes = \{t_0, v_0\}$, $varyings = \{t_1, v_1\}$.

D. $uniforms = \{t_0, t_1\}$, $attributes = \{\mathbf{p}, \mathbf{v}_0.\mathbf{v}_1\}$.

15. A WebGL application starts by generating thousands of randomly placed points in 2D. After that, at each frame, the application moves some points towards the location of the mouse with a speed that depends on the distance to current mouse pointer. The further away, the higher the speed (up to the maximum speed value of that particular point). But not all points are moved at each frame. Each particle is randomly assigned a sequence number in a range [0,N-1] and they only move if the frame number modulo N (the remainder of the integer division) matches their sequence number. What are the attributes and uniforms that you would choose for such application?

A. The initial position and sequence number are attributes, while the maximum speed, current mouse location, time elapsed since last frame and the frame number are uniforms.

B. The initial position, sequence number and the maximum speed are attributes, while the current mouse location, time elapsed since last frame and the frame number are uniforms.

C. The initial position and sequence number are attributes, while the maximum speed, current mouse location, current time and the frame number are uniforms.

D. The initial position, sequence number and the maximum speed are attributes, while the current mouse location, current time and the frame number are uniforms.

16. Consider the GLSL program presented on the right. It is used by an application that draws blinking stars on the screen. Stars are rendered in bundles with all the stars with the same base color drawn in a single `drawArrays()` call. The javascript side of the application generates all the required buffers during initialization. Each star is given a timestamp that establishes when it should start to appear on screen. The blinking effect is simply recreated through the absolute value of a `sin()` wave. Which of the following sequences can be used to fill the blanks in the code so that the application works as described?

```
// vertex shader
attribute ____ v;
attribute float y;
_______ ____ c;
uniform float t;
_______ ____ x;
main()
{
  float o;
  if(t < y) o = 0.0;
  else o = abs(sin(t-y));
  gl_Position = vec4(v, 0.0, ____);
  x = vec4(c, o);
}

// fragment shader
_______ ____ x;
main()
{
  gl_FragColor = x;
}
```

A. `vec3, attribute, vec3, uniform, vec4, 0.0, uniform, vec4`

B. `vec2, uniform, vec3, varying, vec4, 0.0, varying, vec4`

C. `vec2, uniform, vec3, varying, vec4, 1.0, varying, vec4`

D. `vec3, attribute, vec3, uniform, vec4, 1.0, uniform, vec4`

17. Which of the following elementary transformations are not rigid-body transformations?

A. rotation and scaling B. translation and rotation C. shearing and scaling D. shearing and rotation

18. Which of the following composition of transformations can be considered the inverse transformation of:

$$S(2,1,3)\cdot R_z(-30°)\cdot S(2,2,1)\cdot T(2,-4,2)$$

A. $S(1/4,1/2,1/3)\cdot R_z(30°)\cdot T(-2,4,-2)$

B. $S(1/2,1,1/3)\cdot R_z(30°)\cdot S(1/2,1/2,1)\cdot T(-2,4,-2)$

C. $T(-2,4,-2)\cdot R_z(30°)\cdot S(1/4,1/2,1/3)$

D. none of others

19. The Modelview transformation matrix is commonly used in the implementation of 3D graphics systems. Choose the option that correctly describes what it does.

A. It transforms points and vectors from Object coordinates to World coordinates.

B. It transforms points and vectors from World coordinates to Camera coordinates.

C. It transforms points from World coordinates to Camera coordinates.

D. It transforms points from Object coordinates to Camera coordinates.

20. Considering that M_{view} is the matrix returned by a call to setup the camera in a scene using the `lookAt()` function, what is the expression that gives the location, in World Coordinates (WC), of a point located 10 units to the right of the camera location and 5 units in front of the camera location?

A. $M_{view}[10,0,5,1]^T$ B. $M_{view}^{-1}[10,0,-5,1]^T$ C. $M_{view}[10,0,-5,1]^T$ D. $M_{view}^{-1}[10,0,5,1]^T$

A computer screen with UHD-1 resolution (3840×2160 pixels [16:9]) is going to be used to display the contents of a window defined in 2D world coordinates by its limits $-200\le x\le -40$ and $400\le y\le 500$. The viewport will be aligned with the bottom right corner of the screen and it should maximize the viewing area without clipping the window contents or deforming its contents. As usual, the origin of the 2D coordinate system associated with the device has its origin located in the top left corner of the screen.

21. What are the viewport dimensions in pixels?

A. $(\frac{160\times2160}{100})\times2160$ B. $3840\times(\frac{160}{100\times3840})$ C. $\frac{100}{160\times2160}\times2160$ D. $3840\times(\frac{100\times3840}{160})$

22. What would you choose as the first operation to be performed by the window to viewport transformation?

A. $T(-40,400)$ B. $T(40,-400)$ C. $T(3840,2160)$ D. $T(-3840,-2160)$

23. What would you choose as the last operation to be performed by the window to viewport transformation?

A. $T(-40,400)$ B. $T(3840,2160)$ C. $T(-3840,-2160)$ D. $T(40,-400)$

24. What is the scaling transformation used by the window to viewport transformation?

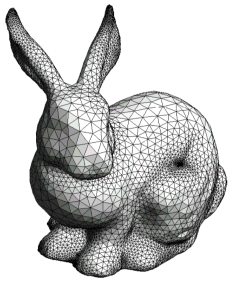
A. S(2160/100, -2160/100) B. S(3840/160, 3840/160) C. S(2840/160, -3840/160) D. S(2160/100, 2160/100)

Consider a 3D scene where the objects do not intersect each other.

25. For such a scene, the visible surfaces determination problem can be solved using the painter's algorithm. The objects are sorted using the order of decreasing distance to the camera. So, objects further away are drawn first and objects closer are drawn last.What can be said regarding the efficiency performance while using the drawing order mentioned above in combination with the Z-buffer algorithm? Choose the correct option!

A. It would be the less efficient order
B. It would be the most efficient order
C. It would have no impact on the efficiency of rendering
D. Nothing could be concluded regarding drawing efficiency

26. Suppose that the objects to be drawn are multiple instances of the Stanford Bunny. Which hidden surface removal techniques should be applied to such a scene in order to solve the problem **in the most efficient way**? Please note that in this question we do not say anyhing regarding the order by which the objects are being drawn.



A. z-buffer
B. z-buffer and backface culling
C. either z-buffer or backface culling
D. back-face culling

Boa sorte!