

Student Number:

Name:

Write your answers in the spaces provided between questions.
Answers written on the back of a page will not be graded.
Write your student number and name on each page.

1. (1 pt each) True / False questions:

- 1) The visible color of an object depends only on the object's material properties and is independent of the color of the light source. ()
- 2) Gimbal lock occurs when two of the three rotation axes in Euler angles align, causing a loss of one degree of freedom. ()
- 3) For texture mapping in OpenGL, GL_NEAREST filtering always produces smoother images than GL_LINEAR filtering. ()
- 4) de Casteljau's algorithm can be used both to evaluate a point on a Bezier curve and to subdivide the curve into smaller segments. ()

2. (3 pts) Which of the following correctly describes the role of the ambient term in the Phong illumination model?

- 1) It simulates direct light reflected off smooth surfaces.
- 2) It adds a constant color to approximate indirect light from the environment.
- 3) It sharpens the specular highlight for shiny surfaces.
- 4) It is used only when normal mapping is applied.

3. (4 pts) Choose **ALL** correct statements about 3D rotation representations.

- 1) Euler angles are intuitive, but may suffer from gimbal lock and can produce undesirable interpolation results.
- 2) Unit quaternions support smooth interpolation but are less intuitive to visualize than Euler angles.
- 3) Rotation matrices use the minimum number of parameters required to represent rotations and are therefore ideal for interpolation.
- 4) Rotation vectors represent rotations with three parameters and can be obtained from rotation matrices using logarithmic mapping.

4. (3 pts) Write down the degrees of freedom (DOF) of each of the following movements:

- 1) Translation on a 2D plane
- 2) Rotation in 3D space
- 3) Any rigid motion in 3D space

Student Number:

Name:

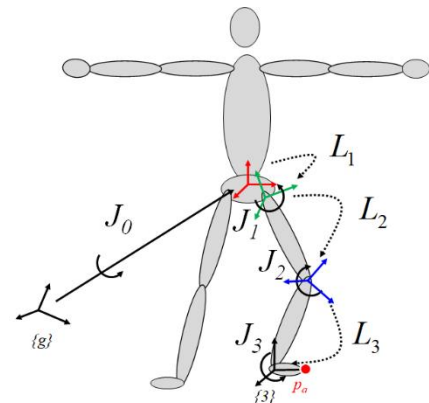
5. (4 pts) Let $\mathbf{R}$ be a rotation matrix in an n -dimensional space. Choose **ALL** statements that are always true.

- 1) $\mathbf{R}^T \mathbf{R} = \mathbf{I}$
- 2) $\det(\mathbf{R}) = 1$
- 3) $\mathbf{R}^{-1} = \mathbf{R}^T$
- 4) $\mathbf{R}$ has three rows and three columns.
- 5) All entries of $\mathbf{R}$ are positive.
- 6) Rotation by $\mathbf{R}$ preserves vector lengths.

6. (4 pts) Choose **ALL false (incorrect)** statement about computer animation.

- 1) In keyframe animation, animators specify important poses at keyframes and the computer generates intermediate frames.
- 2) In keyframe animation, motion is controlled by setting keyframes and manipulating parameter curves.
- 3) It is easy and intuitive to create realistic and physically plausible motion using keyframe animation methods.
- 4) Motion capture systems capture the movement of people or objects.
- 5) Motion capture systems eliminate the need for any post-processing or manual editing.
- 6) Physically based simulation can be used to generate high-quality computer animation because physical laws play an important role in realistic motion.

7. (3 pts) What is the position of point $\mathbf{p}_a$ w.r.t. global frame $\{g\}$, $\mathbf{p}_a^{\{g\}}$? Use the joint and link transformations shown in the figure and let $\mathbf{p}_a^{\{3\}}$ denote the position of $\mathbf{p}_a$ w.r.t. the frame $\{3\}$.



8. (3 pts) Write down the lowest-degree polynomial $x(t)$ that passes through 3 data points $x(0)=1$, $x(1)=-2$, and $x(2)=3$.

Student Number:

Name:

9. (4 pts) Suppose a BVH file contains 'CHANNELS 3 Zrotation Xrotation Yrotation'. Choose **ALL** correct statements.

- 1) Rotations are applied in the order $Z \rightarrow X \rightarrow Y$.
- 2) Changing the CHANNELS line to 'CHANNELS 3 Xrotation Yrotation Zrotation' while keeping the same motion data values always produces the same orientation.
- 3) Rotation order affects the final orientation.
- 4) The rotation values in the motion data are expressed in radians.
- 5) Quaternions are stored directly in BVH files.

10. (4 pts) Let p_0 , p_1 , p_2 , and p_3 denote the first, second, third, and fourth control points of a cubic Bezier curve, respectively.

```
g_control_points = [  
    p0, p1, p2, p3  
]
```

Suppose the following line is executed every frame:

```
g_control_points[1].y += 100*sin(t)
```

Choose **ALL** correct statements.

- 1) The shape of the curve changes over time.
 - 2) One endpoint of the curve changes over time.
 - 3) The curve remains a cubic Bezier curve.
 - 4) The curve always lies inside the convex hull of its four control points.
 - 5) The curve passes through all four control points.
11. (4 pts) Choose **ALL** correct statements about cubic Hermite curves.
- 1) A cubic Hermite curve is determined by two endpoints and two tangent vectors.
 - 2) Changing a tangent vector can change the shape of the curve without changing the endpoints.
 - 3) A cubic Hermite curve always passes through its endpoints.
 - 4) A cubic Hermite curve always passes through its tangent vectors.
 - 5) A cubic Hermite curve is a cubic polynomial curve.
12. (3 pts) What does BRDF stand for in computer graphics?
- 1) Bidirectional Reflection Distribution Function
 - 2) Bioluminescent Reflective Diffraction Function

Student Number:

Name:

- 3) Binary Radiance Diffusion Factor
- 4) Brightness Resilience and Deflection Framework

13. (4 pts) Choose **ALL false (incorrect)** statements about local and global illumination models.

- 1) Local illumination models only consider light that comes directly from light sources, while global illumination models simulate indirect light interactions between object surfaces.
- 2) Local illumination models simulate all light interactions within a limited area, while global illumination models consider the entire scene.
- 3) Local illumination models are usually faster to compute than global illumination models.
- 4) Local illumination models only consider diffuse reflection, while global illumination models consider both diffuse and specular reflection.

14. (4 pts) Choose **ALL** techniques that are used to simulate high-frequency surface by creating the “illusion” of depth without modifying the actual geometry of a 3D model.

- 1) Normal mapping 2) Bump mapping 3) Displacement mapping 4) Environment mapping

15. (3 pts) Match each term about the visibility problem on the left with the corresponding description on the right.

- | | |
|---------------------------|---|
| 1) Clipping | a) Removing back-facing primitives |
| 2) Hidden surface removal | b) Removing primitives outside the viewing frustum |
| 3) Back-face culling | c) Removing primitives occluded by other objects closer to the camera |

16. (4 pts) Suppose an OBJ file contains a quad face ‘f 1 2 3 4’ and your renderer only supports triangles. Choose **ALL** correct statements.

- 1) Vertex duplication cannot occur when converting face data into a renderable form.
- 2) The quad can be rendered by splitting it into two triangles.
- 3) Some vertices of the original quad may be used by multiple triangles after triangulation.
- 4) OpenGL core profile directly supports rendering arbitrary polygons without triangulation.
- 5) A mesh containing both triangles and quads can still be rendered after appropriate preprocessing.

17. (4 pts) Suppose a BVH file contains 1000 frames and the current frame index is updated every frame as

`frame = (frame + 1) % num_frames`

Choose **ALL** correct statements.

- 1) The playback speed depends only on the number of frames in the BVH file.

Student Number:

Name:

- 2) The BVH hierarchy changes every frame.
 - 3) The motion repeats after the last frame.
 - 4) The frame index always stays between 0 and num_frames-1.
 - 5) Removing the modulo operation may eventually cause an out-of-range access.
18. (5 pts) The following code uses the same hierarchical modeling framework as in Lab 10.

```
class Node:
    def __init__( self, parent, link_transform_from_parent,
shape_transform, color):
        self.parent = parent
        self.children = []
        if parent is not None:
            parent.children.append(self)

        self.link_transform_from_parent = link_transform_from_parent
        self.joint_transform = glm.mat4()
        self.global_transform = glm.mat4()
        self.shape_transform = shape_transform
        self.color = color

    def set_joint_transform(self, joint_transform):
        self.joint_transform = joint_transform

    def update_tree_global_transform(self):
        ...
```

The following code creates a simple two-part hierarchical model.

```
def main():
    ...
    base = Node(None,
        glm.mat4(), glm.scale((.2,.2,1.)), glm.vec3(0,0,1))
    arm = Node(base, glm.translate(glm.vec3(.2,0,0)),
        glm.translate((.5,0,.01)) * glm.scale((.5,.1,1.)),
        glm.vec3(1,0,0))
    while not glfwWindowShouldClose(window):
        t = glfwGetTime()
        arm.set_joint_transform(glm.rotate(t, (0,0,1)))
```

Choose **ALL** correct statements.

- 1) Changing arm.shape_transform changes the position of the arm joint.
- 2) Changing arm.link_transform_from_parent changes the offset from the base, so it changes the skeleton structure.
- 3) Changing arm.joint_transform changes the pose or motion of the arm.
- 4) If arm.link_transform_from_parent is changed from glm.translate((1.0,0.0,0.0)) to glm.translate((0.5,0.0,0.0)), the arm joint moves farther from the base.
- 5) If arm.set_joint_transform(glm.rotate(t, (0,0,1))) is removed, the arm no longer has time-varying rotation.

Student Number:

Name:

19. (4 pts) Consider the following texture-mapped object. In the vertex shader, the texture coordinate is modified as follows:

```
uniform float t;

void main()
{
    ...
    vout_uv = vin_uv;
    vout_uv.s -= t;
}
```

In the OpenGL code, the wrapping mode for the s-coordinate is set as follows:

```
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S,
GL_MIRRORED_REPEAT)
```

Choose ALL correct statements.

- 1) The texture appears to move horizontally over time.
 - 2) The geometry of the object moves horizontally over time.
 - 3) Since `vout_uv.s` can go outside the range `[0,1]`, the texture wrapping mode affects the rendered result.
 - 4) With `GL_MIRRORED_REPEAT`, the texture pattern repeats while being mirrored every other interval.
 - 5) If `GL_TEXTURE_WRAP_S` is changed to `GL_CLAMP_TO_EDGE`, texture coordinates outside `[0,1]` are mirrored.
20. (6 pts) Below is the pseudo code for z-buffer (depth buffer) algorithm. Fill in the blanks (a), (b), and (c). You have to use functions already used in the code and variables already defined in the code. Additionally, you can use a function call `read_depth_buffer(x, y)` to get the current recorded depth value at `(x,y)`.

```
allocate depth_buffer;

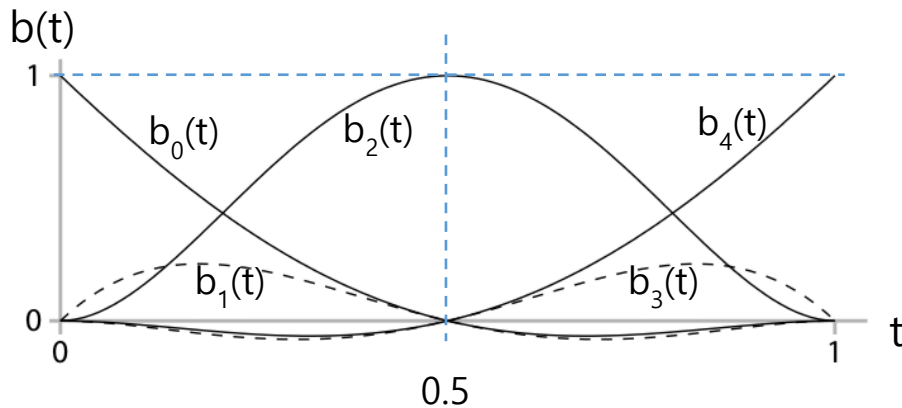
for each pixel (x,y)
    write_frame_buffer(x,y,backgrnd_color);
    write_depth_buffer(x,y,farPlane_depth);

    for each polygon
        for each pixel (x,y) in polygon
            new_color = polygon's color at (x,y);
            new_depth = polygon's z-value at (x,y);
            if ( _____ (a) _____ )
                _____ (b) _____;
                _____ (c) _____;
```

Student Number:

Name:

21. (5 pts) Let's say we have some "unfamiliar" quartic (fourth degree) polynomial curve, which has five basis functions $b_0(t)$, $b_1(t)$, $b_2(t)$, $b_3(t)$, $b_4(t)$ and corresponding control points $\mathbf{p}_0$, $\mathbf{p}_1$, $\mathbf{p}_2$, $\mathbf{p}_3$, $\mathbf{p}_4$. The following figure shows the basis functions (black solid and dashed curves are the basis functions):



- 1) Write down the curve equations $\mathbf{p}(t)$ in parametric form, formulating with given basis functions and control points.
 - 2) Which of the five control points $\mathbf{p}_0$, $\mathbf{p}_1$, $\mathbf{p}_2$, $\mathbf{p}_3$, $\mathbf{p}_4$ will the curve pass through, and for what values of t ?
22. (6 pts) The following code implements matrix-based slerp between two orientations $\mathbf{R}_1$ and $\mathbf{R}_2$. Fill in the blanks (a)-(c).

```
import glm

def slerp(R1, R2, t):
    return R1 * exp( t * log( (a) ) )

eps = 1e-6
def exp(rotvec):
    angle = glm.l2Norm(rotvec)
    if angle > eps:
        axis = (b)
        return glm.mat3(glm.rotate(angle, axis))
    else:
        return (c)

def log(rotmat):
    quat = glm.quat(rotmat)
    return glm.angle(quat) * glm.axis(quat)
```

Student Number:

Name:

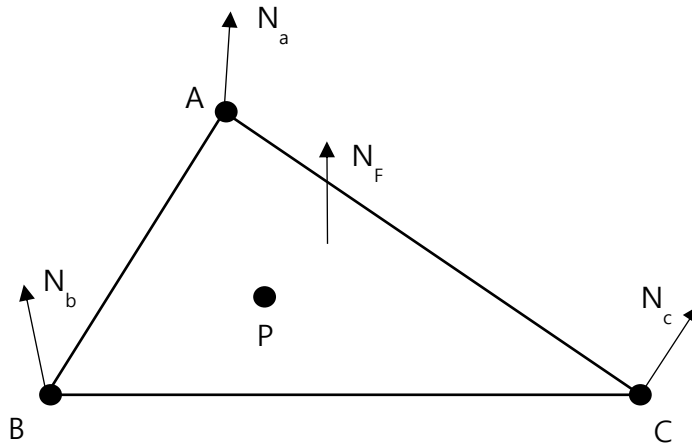
23. (6 pts) The following code is part of a vertex shader that computes the diffuse component of the Phong illumination model for Gouraud shading. Assume that M may include non-uniform scaling. Find and correct all the mistakes in lines a) ~ e).

```
#version 330 core
layout (location = 0) in vec3 vin_pos;    // vertex position
layout (location = 1) in vec3 vin_normal; // vertex normal
uniform mat4 MVP;
uniform mat4 M; // modeling matrix
...
void main()
{
    ...
    // assume that light_pos, light_diffuse, material_diffuse are
    // correctly defined and set
    vec3 normal = normalize(mat3(M) * vin_normal); // a)
    vec3 surface_pos = vec3(M * vec4(vin_pos, 1.0)); // b)
    vec3 light_dir = normalize(surface_pos - light_pos); // c)
    float diff = max(dot(normal, light_dir), 0); // d)
    vec3 diffuse = diff * light_diffuse * material_diffuse; // e)
    ...
}
```

Student Number:

Name:

24. (6 pts) You're about to render a triangle ABC which is one of the triangles in a triangle mesh model. N_a , N_b , N_c are the vertex normals at vertices A, B, and C and calculated as the average of the face normals of the neighboring triangles. N_F is the face normal of triangle ABC.



The illumination model used for this rendering is given as a function $\text{illum}(p, n)$ where p is the position of the target point to be illuminated and n is the surface normal at p to be used for illumination. For example, the illuminated color at the vertex A with N_a as surface normal is $\text{illum}(A, N_a)$.

Question: What is the illuminated color at the point P in the triangle ABC when using...

- 1) Flat shading
- 2) Gouraud shading
- 3) Phong shading

Note that:

- * The barycentric coordinate of the interior point P is given as (u, v, w) , meaning that $P = uA + vB + wC$. Use u , v , and w to compute interpolated color or normal at P.
- * Assume that the interpolated normal is normalized before illumination.
- * A, B, C, and P denote both the points and their positions.