BROCK UNIVERSITY

Final exam
Course: COSC 3P98 Computer Graphics
Date of exam: Wednesday December 11, 2013
Time of exam: 1200-1500

NAME (print): ___________________________________
STUDENT NUMBER: ________________________________

There are 6 questions totaling 164 marks.
One 3” by 5” card of hand-written, non-machine-duplicated notes is permitted.
No other aids are permitted. Use or possession of unauthorized materials will automatically
result in a grade of zero for this examination.
Please answer all questions on the exam paper. Use the back of pages if necessary.
Keep written answers brief and to the point. Write neatly and legibly.
A grade of 40% is required on this exam to pass the course.

<table>
<thead>
<tr>
<th>Question</th>
<th>Total</th>
<th>Mark</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>24</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>24</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>46</td>
<td></td>
</tr>
<tr>
<td><strong>Total:</strong></td>
<td><strong>164</strong></td>
<td></td>
</tr>
</tbody>
</table>
Question 1 [24] Multiple choice: select the best answer to each question
[+2 correct, 0 nothing, -0.5 wrong guess; one free wrong guess without penalty]

1. Which of the following is not a basic transformation:
   a. rotate
   b. translate
   c. perspective
   d. scale

2. The illumination effect responsible for "shiny surface details" of objects is:
   a. specular
   b. ambience
   c. diffuse
   d. refraction

3. OpenGL’s alpha channel is used to:
   a. keep track of nearest object at each pixel
   b. compute forward-facing normals
   c. save material coefficients
   d. give illusion of transparency

4. The OpenGL command that defines a 3D vertex with an array “a” of unsigned bytes is:
   a. glVertex3ubv(a)
   b. glBegin(a[1], a[2], a[3])
   c. glVertex3ub(a[1], a[2], a[3])
   d. glVertex(a)

5. Linear interpolation uses:
   a. curved changes between 2 points
   b. even-spaced changes between 2 points
   c. keyframes
   d. minimal weight changes between 2 points

6. A simple example of a procedural 3D texture is:
   a. map XYZ coordinate to RGB value
   b. bump maps
   c. wrap a bitmap onto polygon surface
   d. apply anti-aliasing to colour rendering
Question 1. (cont)

7. The advantage of the midpoint line drawing algorithm is:
   a. it uses high-resolution floating-point arithmetic
   b. it reduces jagged stair-casing effects compared to other algorithms
   c. it uses fast integer arithmetic
   d. none of the above

8. Telling GLUT which user function to use during a keyboard event is also known as:
   a. callback registration
   b. immediate-mode graphics
   c. RGB-mode colour
   d. display list processing

9. A defining characteristic of minimal-weight triangulations is:
   a. minimal number of triangles
   b. the surrounding edges form a hull
   c. the total area of all the triangles is minimized
   d. the total length of all the edges is minimized

10. The basic transformations are:
    a. transformational
    b. orthographic
    c. affine
    d. awesome

11. An uncompressed full-colour RGB 640-by-480 image uses approximately this much memory:
    a. 307200 bytes
    b. 102400 bytes
    c. 921600 bytes
    d. 1228800 bytes

12. The perspective projection’s clipping volume is known as a:
    a. Rob-Ford-oid
    b. gamut
    c. frustum
    d. convex hull
Question 2 [20] Define and briefly discuss the following terms or OpenGL /GLUT commands:

a) glutIdleFunc

b) orthographic projection

c) ambient lighting

d) tweening

e) glPushMatrix
Question 3 [24]

(a) [12] Derive a general homogeneous 2D transformation matrix for the following 2D transformation: scale an object with factors (SX, SY) such that an object stays locked at its location, and does not move horizontally or vertically after the scaling. (Hint: let (VX, VY) be a coordinate of one of the vertices on the object)).
Question 3 (cont)

(b) [12] Give an example using real homogeneous matrices and 2D transformations that shows why the ordering of transformations matters. You can use actual values in the matrices. Include an illustrative diagram as well, and explain your steps.
Question 4 [30]

(a) [15] Explain the basic steps used in performing Gouraud shading. Include a diagram. Include in the discussion the pertinent place where the lighting model discussed in class is included in the Gouraud process. Also list 2 advantages and 2 disadvantages of Gouraud shading.
4 (b) [15] Explain how Phong shading works. Include a diagram, and explain all the pertinent points. Also include where the lighting model comes into consideration. Identify how Phong is an improvement over Gouraud shading. Also identify one disadvantage of it.
Question 5 [20]

Give pseudo-code for the naive (greedy) triangulation algorithm. You only need to discuss the edge creation phase; you can ignore the polygon construction phase. Discuss what the algorithm steps are doing. Include a diagram to help your discussion. (Note: you do not need to include the algebraic formulae used; however, do describe the important calculations required at various steps).
Question 5 (cont)
Question 6 [46]

Write C/OpenGL/GLUT code that does the following. Read the entire question before beginning. Question parts rely on the use of others. Please do not implement anything that is not requested (ie. user commands, lighting, textures, menus, etc.). Use the backs of pages if necessary.

(a) [8] Define some C data structures that can save the following:

- 100 triangles (vertex ranges 0.0 ≤ x, y, z ≤ 1.0)
- 100 (possibly non-planar) 4-vertex polygons (“4-poly”) (same range as triangles)
- an RGB value per triangle and 4-poly (1 colour each)
- a random translation factor per triangle or 4-poly (-1000.0 ≤ dx, dy, dz ≤ 1000.0)
(b) [12] Write a function that fills the triangle and 4-poly data structures defined in (a) with random values, appropriate to the ranges specified. Presume you can call random() to generate a random float between 0.0 and 1.0.
(c) [20] Write a callback which does the following. The frame is cleared, and then the entire data structure for either triangles or 4-polys is rendered in the world. You can presume that GLUT and OpenGL have been appropriately initialized for this purpose.

A global integer variable "polyType" is available. If polyType has the value 0, then the triangles are rendered as solid polygons. Otherwise, the 4-polys are drawn. Use transformations to aid in the rendering of the polygons. Make sure you include the colour information.

The callback should render one frame, and return control to GLUT. Repeated calls to the callback should be able to be used in a smooth double-buffered 3D animation (perhaps the world is rotating, in a manner like "rotate2.c" used in class?).
(c) cont.
(d) [6] Note that the 4-polys as described above will probably not be planar, because the random vertices will not usually be on the same plane. (Triangles, however, are always planar). Can you think of a way in which random-sized planar rectangles could be defined, such that they have arbitrary orientations in the 3D world as defined above?

Do NOT include complete C-code for your answer. Rather, just describe how you could accomplish this task! Feel free to supplement data structures if required. (C-code snippets or pseudo-code can be optionally included, if they help your discussion).