Stanford Honor Code Statement:

The Honor Code is an undertaking of the students, individually and collectively:

1. that they will not give or receive aid in examinations; that they will not give or receive impermissible aid in class work, in the preparation of reports, or in any other work that is to be used by the instructor as the basis of grading;

2. that they will do their share and take an active part in seeing to it that others as well as themselves uphold the spirit and letter of the Honor Code;

The faculty on its part manifests its confidence in the honor of its students by refraining from proctoring examinations and from taking unusual and unreasonable precautions to prevent the forms of dishonesty mentioned above. The faculty will also avoid, as far as practicable, academic procedures that create temptations to violate the Honor Code.

While the faculty alone has the right and obligation to set academic requirements, the students and faculty will work together to establish optimal conditions for honorable academic work.

I acknowledge and accept the Honor Code.

Signature: _____

Name (Print): _____

Date: _____

CS148 Final, Summer 2010

You have 2 hours and 15 minutes to complete this exam
The exam is closed-everything, including calculators

The question paper has 12 pages, 5 questions and 50 points (including the page for the Honor Code statement and the extra credit question). Avoid attaching extra sheets as far as possible. You may use blank scratch sheets, but don't submit them. You don't need to show steps in a solution unless they're explicitly asked for.

NAME: _____

1. Light, Color, Cameras and Images *(12.5 points)*

1a) Recall that CMYK separation is not unique, i.e. the same image can be broken down into cyan, magenta, yellow and black channels in more than one way.

i) How many ways are there to do this decomposition? *Unlimited* *(1 point)*

ii) For reproduction on a typical inkjet printer, would you prefer the separation that has the maximum black component, or minimum black component? Explain your answer. *(1 point)*

Maximum black, because mixing CMY to make black usually results in a muddy shade, not true black. We also accepted arguments based on cost etc.

1b) The ProPhoto RGB color space has a larger gamut than sRGB. Graham owns a camera capable of capturing images covering the entire ProPhoto RGB gamut, but his neighbor John's camera only captures the sRGB range. The cameras are otherwise identical. In the local photo exhibition, images from both cameras will be displayed on the *same* sRGB display. Does Graham's camera have any advantage over John's for this application? Justify your answer. (You may assume the pixel values are represented with arbitrarily precise real numbers in each case, so precision is not a differentiating factor.) *(2 points)*

Graham's camera actually does have an advantage. If the scene being photographed has tonal variations in the part of the ProPhoto RGB gamut outside sRGB, then this can be captured and mapped to nearby sRGB shades for display. Thus one can see the detail even though the colors displayed are not exactly accurate. John's camera has no way of capturing, and hence representing, such detail.

1c) Terry spends a week processing a JPEG image. Each day, he loads the image into Photoshop, makes some minor edits, and then resaves the image as JPEG. At the end of each day, he quits Photoshop and shuts down his computer. Why is this a bad workflow? How could Terry do better? Explain clearly why your approach would improve matters. *(2 points)*

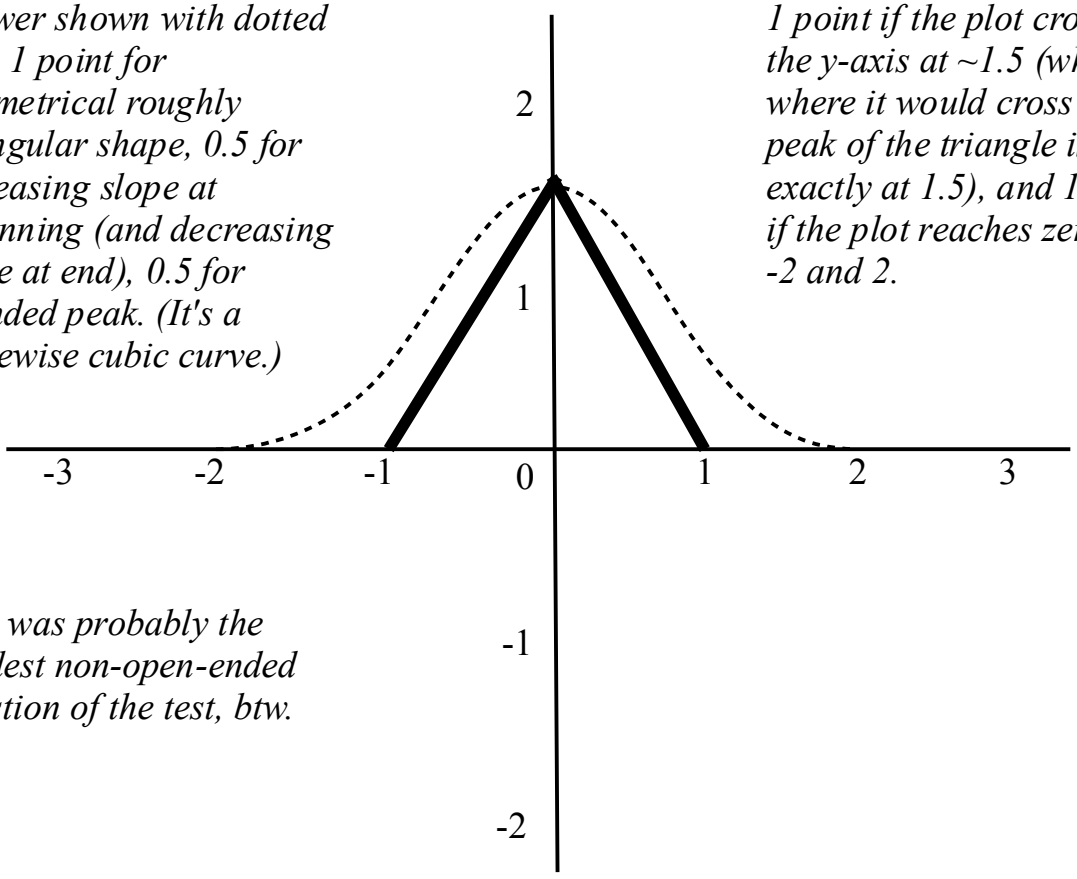
JPEG is a lossy compression scheme, so every time Terry saves the image he loses some data. The errors accumulate over time and can severely degrade the image. Terry should save his working copy in a lossless format (e.g. PNG or TIFF) at every stage of his workflow.

Those of you who suggested never shutting down the computer, so the image is held uncompressed in memory throughout, should really be given zero (think of the potential to lose all your work if the machine crashes or there's a powercut, not to mention the huge waste of power unless the machine is set to hibernate!) But we were feeling generous so I think you got credit anyway.

1d) Draw the convolution of the following triangular function with itself. Your sketch should be on the same graph, and roughly to scale. *(2 points for the correct basic shape + 2 points if the plot crosses the axes at approximately the correct coordinates)*

Answer shown with dotted line. 1 point for symmetrical roughly triangular shape, 0.5 for increasing slope at beginning (and decreasing slope at end), 0.5 for rounded peak. (It's a piecewise cubic curve.)

1 point if the plot crosses the y-axis at ~1.5 (which is where it would cross if the peak of the triangle is exactly at 1.5), and 1 point if the plot reaches zero at -2 and 2.



This was probably the hardest non-open-ended question of the test, btw.

1e) What is the function of the alpha channel in an image? What do alpha = 0 and alpha = 1 correspond to? *(1 point)*

Alpha represents opacity (we also accepted transparency as an answer). Alpha = 0 means the pixel is fully transparent, and alpha = 1 means it's fully opaque.

1f) Recall that in blue screen matting, the actor stands in front of a blue screen and the part of the image corresponding to this screen is automatically removed, to be replaced with a different background. Assume all objects in the scene are opaque, no part of the actor or his/her clothing is blue, and the screen appears an absolutely uniform blue color in the image (no variations in shading). Why is it not sufficient to simply delete all pixels that are approximately the same color as the screen? *(1.5 points)*

Pixels at the edge of a foreground object can be partly covered by the foreground and partly by the background. Such a pixel should not be retained or deleted wholesale, since that would lead to aliased outlines. Instead, the pixel should be made semi-transparent (the more it is covered by the background, the more transparent it is) by setting its alpha value, after filtering out the contribution from the background.

2. Geometry, Curves and Surfaces *(12.5 points)*

2a) Vectors interpreted as positions and directions are different. Which of these operations are legal, and which are illegal? (Tick/check one in each case.) Consider both arguments and return type. *(8 * 0.5 = 4 points)*

Position = Scalar * Position	Legal _____	Illegal ✓
Direction = Position – Position	Legal ✓	Illegal _____
Position = Position + Direction	Legal ✓	Illegal _____
Direction = Direction + Direction	Legal ✓	Illegal _____
Position = Position – Position	Legal _____	Illegal ✓
Position = Position + Position	Legal _____	Illegal ✓
Direction = Position + Direction	Legal _____	Illegal ✓
Direction = Scalar * Direction	Legal ✓	Illegal _____

2b) Find a formula for the normal to the implicit 2D surface

$$ax^3 + bxyz + cz^2 + 1 = 0$$

embedded in 3D, at a point (x, y, z). Briefly outline how you arrived at your answer. (Your answer should be in terms of the variables x, y and z. We'll accept either of the two valid, opposing directions for the normal. The normal need not have unit length.) *(2 points)*

Recall that the normal to an implicit surface of this form is $\left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}\right)$ *, where f is the function equated to zero (the LHS here). Differentiating w.r.t. each of x, y and z respectively, while holding the other two variables constant, we obtain the normal as*

$$(3ax^2 + byz, \quad bxz, \quad bxy + 2cz)$$

2c) A curved surface may be approximated in piecewise linear fashion by a polygon mesh. This process is called tessellation. A mesh consists of vertices, edges and faces.

i) Name a rendering trick to make a mesh appear to be a smoothly varying surface. (A description of such a technique is not a valid answer.) *(1 point)*

Phong shading and Gouraud shading are both valid answers. We also accepted smooth subdivision schemes like Catmull-Clark, though this wasn't exactly what we were looking for since it modifies the mesh.

ii) Eric writes a procedure to tessellate an arbitrary curved surface. The vertices of the resulting mesh are guaranteed to sample the curved surface uniformly by area. When is this a good thing, and when is this a bad thing? Explain your answer. *(1 + 1 points)*

Good when the mesh has roughly uniform curvature/detail, since there is no reason to over- or under-tessellate any one place. Bad when the mesh is non-uniform, since areas with higher curvature/detail require smaller facets to represent them accurately, but it's wasteful to subdivide the flatter areas to the same level. Varying the tessellation according to the amount of local detail is called "adaptive tessellation".

Some of you stated that it's bad for meshes with high curvature. That's not quite true. The problem is not high curvature (the method could just give uniformly high tessellation in this case) but **variation** *in curvature.*

2d) For two curve segments to be joined together, define: *(1 + 1 points)*

C^0-continuity: *The sections meet end-to-end.*

C^1-continuity: *C^0 + The sections have the same first derivative/tangent at the junction.*

2e) What is the principal difference between cubic Hermite splines and cubic Bézier splines (other than the mathematical descriptions, of course)? *(1.5 points)*

Hermite splines are guaranteed to pass through all the control points. Depending on how you join up the segments, Bézier splines (in general) only pass through the first and last control point of each segment, or through only the very first and very last control point.

This is why curves like Hermite splines are called "interpolating splines" and curves like Bézier splines are called "approximating splines".

3. Rendering and GPGPU *(12.5 points)*

3a) GPUs are special-purpose processors designed to display lots of geometry very fast. Briefly answer the following questions about GPUs.

i) Why are scalar (serial processing) chips not getting significantly faster? *(1 point)*

The main reason is that it becomes more and more difficult to dissipate the heat generated by increasing clock speed.

ii) Give a characteristic of code that can run very fast on the GPU. *(1 point)*

Many of you suggested that the code should be parallelizable. We gave half-credit for this answer. It's not enough that it be parallel – it should be embarrassingly *parallel, meaning that the threads are independent and don't need to communicate. Answers mentioning data-level parallelism, absence of branching etc. got full credit.*

iii) Which of the following would you expect to find in a GPU shader core circa 2010? *(10 * 0.5 = 5 points)*

Fetch-Decode Unit	Yes ✓	No
L3 Cache	Yes	No ✓
L2 Cache *(free 0.5, we accepted either answer since there's no real rule-of-thumb at the moment)*	Yes ✓	No ✓
Several Arithmetic Logic Units (ALUs)	Yes ✓	No
Only a single ALU	Yes	No ✓
Registers	Yes ✓	No
Memory Reading/Writing Units	Yes ✓	No

Memory Pre-Fetcher	Yes _____	No __✓____
Branch Predictor	Yes _____	No __✓____
Double Precision Floating Point Unit	Yes __✓____	No _____

iv) Name an API or language designed specifically for writing GPGPU programs. *(1 point)*

CUDA, OpenCL, Brook etc.

3b) Place these objects in the order in which they are processed in the graphics pipeline (the one that is processed earliest should be placed first): **pixels**, **primitives**, **vertices**, **fragments**. *(2 points)*

Vertices, primitives, fragments, pixels.

3c) Terry writes a z-buffer-based graphics application using OpenGL and executes it on a machine with a modern GPU. Terry's program simulates transparency by setting an alpha value for each fragment. If the fragment passes the depth test, the framebuffer pixel is updated by blending the fragment color with the existing color according to this alpha value. What could go wrong with this scheme? How would you fix it?
(1 point for error in Terry's scheme + 1 point for partial fix + 0.5 for complete fix)

The problem is that multiple fragments at a single pixel location are not guaranteed to arrive in back-to-front order. Out-of-order blending gives incorrect results in general. There are two main solutions, neither of which is truly satisfactory (we gave either one full credit):

- Sort the objects in the scene back-to-front and render in this order. 0.5 reserved for observing that the order might be ambiguous for some pairs of objects (see handouts), and these might need to be split up into smaller subobjects.

- Disable the depth-test, accumulate a sorted list of all fragments at each pixel, and blend (in order) right at the end. 0.5 reserved for observing that not all fragments can be cached (infeasible memory/hardware requirements) so only the nearest k layers should be maintained. This is called a "k-buffer". A variation on this theme renders the scene in multiple passes, identifying the nearest layer of surface fragments first, then the layer immediately behind the first (using the first layer as a strict lower bound for the depth test), then the third layer, and so on. The k layers are merged in a final step. This is called "depth-peeling".

4. Animation, Interaction, Visualization and Typography *(12.5 points)*

4a) Recall that Inverse Kinematics (IK) solves the following problem: Given a few constraints, make a body with many degrees of freedom move according to those constraints. This is usually a very underconstrained problem: the body has far more degrees of freedom than there are constraints. Describe any common way in which an IK system might further constraint the motion to derive a unique solution. *(1 point)*

Possible answers:

- *Minimize the net displacement of each part/point of the body*
- *Train the system to mimic captured motion (e.g. of a human) so that the movement appears plausible.*

Some folks mentioned physics-based simulation, which would certainly work, but we felt this went too far into the domain of dynamics.

4b) Distinguish between polling and interrupts. *(2 points)*

Polling: Application periodically queries system/OS for inputs (synchronous)

Interrupts: OS notifies application whenever input is received (asynchronous)

4c) Michael is studying the distribution of authors of books in his collection. For each author, he has two fields: birthplace (geographical location) and genre (a string). Sketch an effective visualization that simultaneously presents the distribution of both these attributes to Michael. Label your sketch to indicate particular features of your visualization. You may indicate, with text or crosshatching, the use of color if necessary. (This is an open-ended question – there is no one correct answer though some are obviously better than others. You should try to ensure Michael can glean as much information as possible from a quick glance.) *(3 points)*

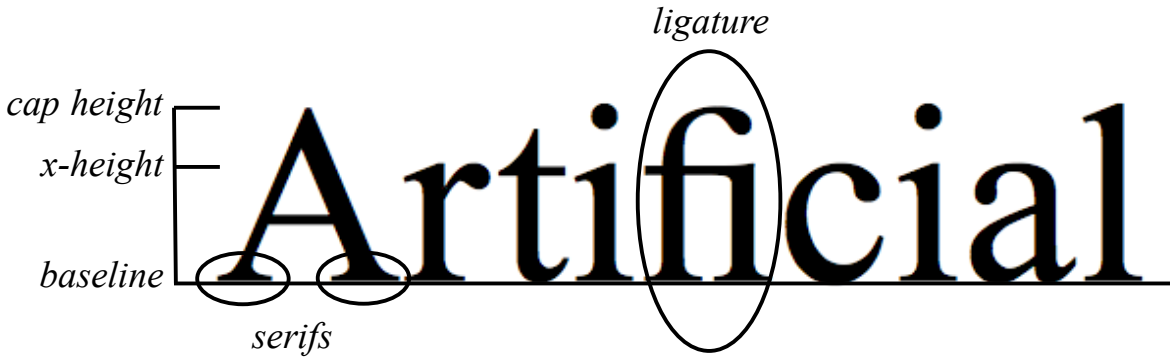
Points on a map, colored by genre, received nearly full credit. We reserved 0.5 points for folks who identified that having multiple authors (possibly of different genres) born at the same location is a problem, and tried to deal with it (e.g. by having larger points to represent clusters of authors, or even pie-charts to show the breakup at a single location).

Placing text on a map is not the best idea – it doesn't scale well and it's not as easy to identify patterns quickly as it is with color.

Note that you didn't have to write the author's name anywhere. This wasn't part of the data Michael had to visualize.

Jason confirms that this is actually quite a hard research problem :). Our suggested solution should not be seen as a definitive answer in any way.

4d) Here is a word typeset in a serif font.



i) Mark and label the cap height, x-height and baseline on the image. *(1.5 points)*
ii) Identify any two serifs (circle them and label them: "serif"). *(1 point)*
iii) Identify a ligature (circle it and label it: "ligature"). *(1 point)*

Please be unambiguous in your labeling, else you won't get credit.

4e) Define the following two terms, in the context of typography. *(1 + 1 points)*

i) Leading: *Distance from the baseline of one line of type to the next (line spacing).*

*(**NOT** the indentation of a line from the margin.)*

ii) Kerning: *Adjusting spacing between letters for better appearance.*

(Note: kerning is the **adjustment***, not the spacing itself.)*

4f) State one way in which Unicode is an improvement over ASCII. *(1 point)*

Possible answers:
- *Much larger character set*
- *Separates representation from representable set*
- *Many encodings*

5. Extra Credit *(0 points)*

Answer one or more of the following:

i) Suggest a cool new assignment for CS148.
ii) What's a good place to eat out within 50 miles of where you are now, which won't bust our meagre grad student incomes? We're pretty familiar with University Ave, so out-of-the-way places are preferred.
iii) Tell us a joke we haven't heard before.

(To clarify: the answer, or lack of one, absolutely does not affect your grade.)