Full Name: ______________________________

Andrew ID: ______________________________

15-462/662, Fall 2018

Midterm Exam

October 17, 2018

Instructions:

- This exam is **closed book, closed notes, closed neighbor, closed cell phone, closed telepathy, closed internet**.
- You may however use a single 3in x 3in sticky note (or piece of paper) with any information you like written on both sides—-*except* for solutions to previous exams.
- If your work gets messy, please clearly indicate your final answer (by writing it in a box if possible).
- Partial credit will be awarded, *but only if we can understand your work!* So please try to write clearly, especially if you are uncertain about the final answer.

Problem	Your Score	Possible Points
1		30
2		25
3		33
4		12
EC		5
Total		100

1 (30 points) Sampling and Aliasing

A. (5 points) In your own words, what is aliasing? How does it relate to sampling and reconstruction?

B. (5 points) Give an example of aliasing found in computer graphics. (This example must be different from the "jagged edges" that occur in line rasterization.)

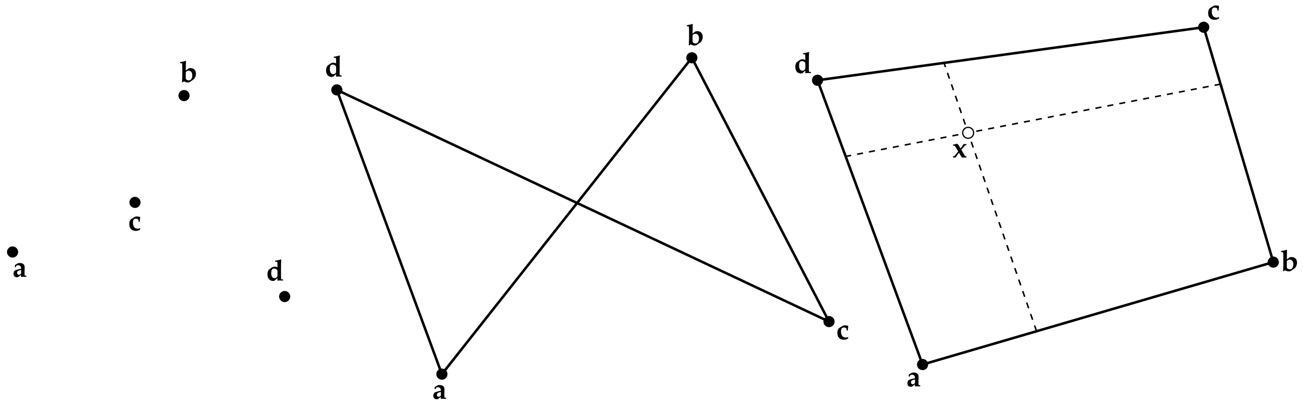
C. (5 points) Give an example of aliasing found *outside* of computer graphics/computer science. (This example must be different from the "wagon wheel" effect described in class.)

D. (5 points) You've been hired to work on the next generation of video distribution software for YouTube. The iPhone XII has just been released, and by default records video at 480Hz. However, having taken 15-462 you know that most video will look good if played back at 60Hz—in fact, the new YouTube player you're designing *only* supports 60Hz playback, independent of the original source rate. Taking both bandwidth and amortized computational effort into account, what is a more appropriate strategy for the software you're writing: *supersampling* or *prefiltering?* Why?

E. (5 points) In the YouTube scenario described in the previous question, why isn't *subsampling* a good strategy? What kinds of videos will look particularly bad if you use subsampling?

F. (5 points) Since you were hired at YouTube, personal assistants like Alexa, Siri, and Cortana have been adding robotic legs and low-quality surveillance cameras. Scary. Since these cameras record video at only 24Hz, how could you generate 60Hz video that could be played back in the YouTube video player? Try to be specific about your strategy. (Deep learning is not allowed.)

2 (25 points) Quadsterization



quadVidia has decided that triangles are no longer cool, and quads are the next big thing[1]. Your graphics consulting company, PinkFongGFX, has been hired to develop the rasterization algorithms for quadVidia's next-generation architecture. The basic setup is that you're given four vertices $\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d} \in \mathbb{R}^2$, in any order. quadVidia has asked that your algorithms correctly rasterize any *convex* quad, but may cull (reject) nonconvex quads, which are not considered to be valid input.

A. (5 points) To simplify assumptions at later stages of the pipeline, the first thing you want to implement is the culling stage. How would you test that the four given points describe a convex quad?

B. (5 points) Now that you know the points define a convex quad, how would you test whether they're in a consistent counter-clockwise order? If they're not, how would you fix the ordering so that they are?

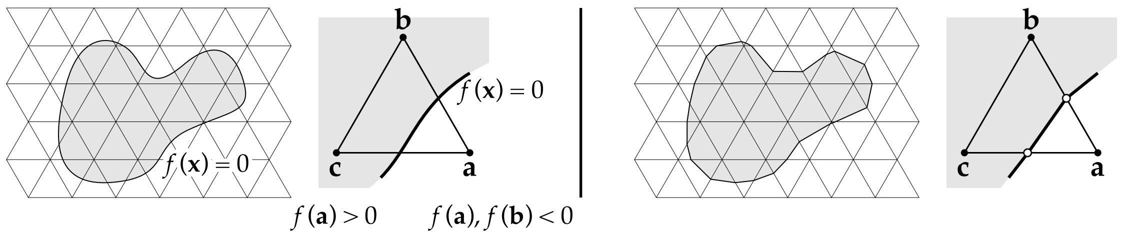
[1]Historical aside: the original NVIDIA graphics chip, the NV1, actually *did* rasterize quads... nobody bought it!

C. (5 points) Now that you know you have a convex quad with points **a**, **b**, **c** and **d** in counter-clockwise order, how would you test that a given sample point x is inside the quad?

D. (5 points) Finally, quadVidia wants your algorithm to provide coordinates for interpolating vertex attributes like colors and texture coordinates. Why might splitting the quad into two triangles and using barycentric interpolation in each triangle be a bad idea?

E. (5 points) Since barycentric coordinates are no good, you've decided to use *bilinear coordinates*, *i.e.*, the coordinates such that **x** is a bilinear interpolation of the four corner vertex coordinates. Write down the system you would have to solve in order to obtain these coordinates. How many equations are there, and how many unknowns? ***Extra credit (5 points):*** *solve it!*

3 (33 points) Converting Geometry



In class we spent a lot of time talking about various explicit and implicit representations of geometry. Of course, no one representation will work well for all algorithms, and so in practice you'll have to convert between the two. Here we'll run through a few exercises where we translate between representations.

A. (5 points) Suppose we're given a curve represented as the zero set of a function $f(\mathbf{x})$, and want to convert this zero set into a polygon. Are each of these representations explicit or implicit, and why?

B. (6 points) One algorithm for approximating a zero level set by a polygon is called *"marching triangles."* The idea is to lay down a grid of triangles over space, and evaluate the function f at each vertex. If the sign of f is different on two neighboring vertices $\mathbf{a}$ and $\mathbf{b}$ (*e.g.*, positive at one end; negative at the other end), then the zero set must cross the edge somewhere between $\mathbf{a}$ and $\mathbf{b}$. In this situation, the marching triangles algorithm inserts a crossing at the location where the linearly interpolated value of f is equal to zero. Write a simple routine that computes the crossing location, given the locations of the two endpoints, and the values of f at the two endpoints. This method should store the solution in $\mathbf{p}$, and return false if and only if there is no crossing along the edge.

```
bool crossing( Vector2D a, Vector 2D b,
                     double fa, double fb,
                     Vector2D& p )
{











}
```

C. (6 points) Now that we know how to find the crossing point for a single edge, we just have to connect up crossings within each triangle. (Think: how many different ways can this happen, given that we're only sampling f at the vertices?) Implement a routine that takes as input the three vertex coordinates $\mathbf{a}, \mathbf{b}, \mathbf{c}$, and the three function values $f(\mathbf{a}), f(\mathbf{b}), f(\mathbf{c})$, and appends any segments passing through this triangle to a list of pairs. You can initialize a pair by writing `P = pair(p,q)`, and you can append a pair `P` to a list by writing `list.append(P)`. Note that your routine should call the `crossing()` method (even if you did not implement it).

```
bool marchTriangle( Vector2D a, Vector2D b, Vector2D c,
                    double fa, double fb, double fc,
                    list<pair>& segments )
{









}
```

D. (5 points) If we run `marchTriangle` on every triangle in our grid, we get a list of segments that make up our polygon. At this point, however, we really just have "segment soup": we don't know which segments are connected to which other segments. What data structure might you use to quickly locate segments with equal endpoints? Why?

E. (5 points) Suppose we want to go the other direction: we have a polygon, and want to compute a signed distance value $f(\mathbf{a}$ for every vertex $\mathbf{a}$ of a regular triangular grid. What data structure would you use to accelerate computation of these values? If there are m total vertices on the grid, and n total vertices on the polygon, what would be the total cost of computing the signed distance function (using your chosen data structure)? You should assume that m is *much* larger than n, and give the amortized asymptotic cost.

F. (6 points) Suppose we start with a function $f(\mathbf{x}) = 0$, run marching triangles to get a polygon, and then sample the distance to this polygon back onto the original grid to get values $f'(\mathbf{a})$ at each vertex $\mathbf{a}$. Are these distance values the same as the original ones? In other words, does $f'(\mathbf{a}) = f(\mathbf{a})$ for all $\mathbf{a}$? What if we run through this whole loop one more time, *i.e.*, we run marching triangles on the function f', then compute the distance f'' to the resulting polygon. Does $f'(\mathbf{a}) = f''(\mathbf{a})$? If there is any loss of information, at which step(s) does it occur, and what name would you give to this phenomenon?

4 (12 points) Transformations

A. (6 points) Of the five sequences of 3D transformations listed below, two pairs are equivalent. Which pairs are the same, and which one is different from the other four?

(a) Rotate by 90 degrees around the z-axis, then translate by a distance 2 along the y-axis, then scale by a factor 2.

(b) Rotate by 90 degrees around the z-axis, then scale by a factor 2, then translate by a distance -2 along the x-axis.

(c) Translate by a distance 2 along the y axis, then rotate by 90 degrees around the z-axis, then scale by a factor 2.

(d) Rotate by -270 degrees around the z-axis, then translate by a distance 2 along the x-axis, then scale by a factor 2, then translate by a distance -8 along the x-axis.

(e) Scale by a factor 2, then rotate by 90 degrees around the z-axis, then translate by a distance 4 along the y-axis.

B. (6 points) Write down a 4×4 matrix that encodes transformation (a) from the previous question in homogeneous coordinates. Apply it to the three points $\mathbf{p} := (0,0,0)$, $\mathbf{q} = (1,0,0)$, $\mathbf{r} := (0,1,0)$, and to the unit normal N of the triangle made by points $\mathbf{p}$, $\mathbf{q}$, and $\mathbf{r}$. You should assume this normal points in the same direction as the vector $(\mathbf{q}-\mathbf{p}) \times (\mathbf{r}-\mathbf{p})$.