CSC 240: Computer Graphics
Midterm Examination: Spring 2021
Due: Thursday, April 1 at 11:59 pm (on Moodle)

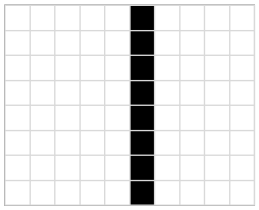
- This is a take-home exam with unlimited time from when it is out to when it is due. I don't expect you to work on it the entire time; it should take you just a few hours.
- It is open-notes, so you may use any course materials. If you use any online resources that haven't been part of this class, please cite them explicitly.
- You may not communicate or consult about the exam with anyone in the class (or outside the class, including requesting help by posting questions on the internet).
- You can post privately on Piazza if you need clarification on any question. If there is a clarification I think should be made to the entire class, I'll make the post public.
- I will still have office hours as usual, but I might not say much about the exam!
- Turn in your exam electronically on Moodle. (You may scan handwritten work.)
- If you are unable to make progress on any part of the exam, tell me what you tried: describe your thought process. I may be able to grant partial credit.
- When your exam is complete, before submitting it, please copy, sign, and date the statement below:
  "I certify that my work on this exam adheres to the Smith Honor Code and the instructions given above. I have explicitly cited any resources used beyond my own notes and the materials available from the course web page."

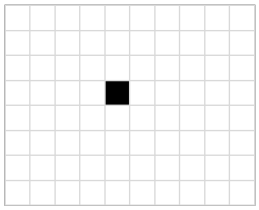
Signed: Date:

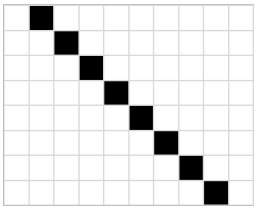
Name:	
Line drawing	/18
Fill algorithms	/16
Transformation matrices	/20
Animation	/10
Clipping	/20
Bézier splines	/16
Total	/100

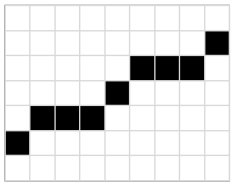
Line Drawing (18 points)

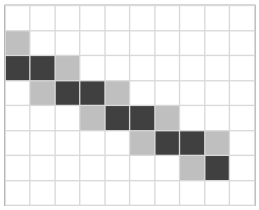
Below are close-up images of portions of a drawing canvas. For each image, decide whether it could have been generated by using each of the algorithms listed below to draw just a single line.

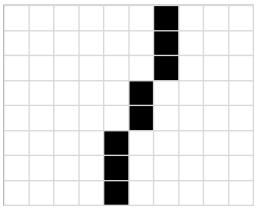
a. 

b. 

c. 

d. 

e. 

f. 

Algorithms: (I.) simple midpoint algorithm, (II.) iterative midpoint algorithm, (III.) Wu's antialiased line.

a: I, II, III. b: I, II, III. c: I, II, III. d: none. e. III only. f. I and II only.

Note that I and II produce similar results, at different speeds. III can produce all black and white pixels if the line is perfectly aligned to the grid. There is no straight line passing through the points of (d).

Fill Algorithms (16 points)

Your friend is excited to learn about fill algorithms, and proposes two new ones for your consideration.

Algorithm A is a recursive algorithm. Instead of using a fixed order for the recursive calls, it always makes eight calls in a clockwise order starting from the square it just came from. So for example, when a recursive call is made to the north, the next calls will first try south, then southwest, then west, etc. If a recursive call is made to the west, the next calls will try east, then southeast, etc. The very first recursive call is always made to the north.

Algorithm B is an iterative sweep algorithm. When called, it will fill the current row by sweeping left then right just as we do in ordinary sweep fill. Then it will loop from the leftmost extent to the rightmost just as in ordinary sweep fill, but instead of making recursive calls it will instead do an iterative vertical sweep, filling up and then down until it hits a boundary. After each vertical sweep, it makes a recursive call to the left and the right at the very top, and again at the very bottom (but not at any pixels in the middle).

Your friend wants to know what you think of these ideas. Please answer for each algorithm:

a.) Will it produce the same end result as the fill algorithms we have studied?
Algorithm A will give the same end result as the usual recursive fill. Algorithm B will sometimes fail to fill the entire shape.

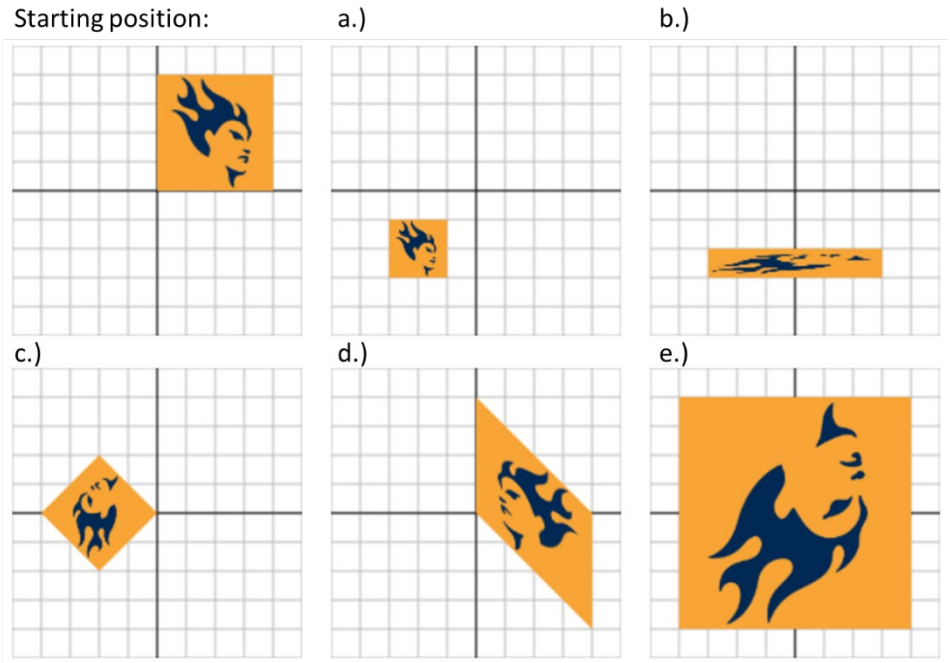
b.) If not, describe the case(s) where the outcome differs. What sort of shapes give a different result?
Shapes with indentations on the left or right sides will not fill completely with Algorithm B.

c.) How many recursive calls would each algorithm make to fill a 100x100 square region?
Algorithm A will make 8 calls per pixel filled, for a total of 80,000 but 70,000 return immediately. Algorithm B will end up making four per row, for a total of 400 but 300 return immediately.

d.) Compare your answer above to the number of recursive calls made by the two algorithms we studied.
Algorithm A will make the same number of recursive calls as the standard 8-connected recursive algorithm, or twice as many as the 4-connected algorithm. Algorithm B will make fewer calls, since our standard sweep fill uses two per square filled (above and below each pixel).

Transformation Matrices (20 points)

Give the 2D transformation matrix that would accomplish each transformation shown below. (Hint: you can compose them out of the basic transformation types we studied.) Use homogeneous coordinates. The grid squares shown are all 1x1, and the Y axis is normal (not inverted as in screen coordinates). The black axis lines cross at the origin.



a.) $\begin{bmatrix} 0.5 & 0 & -3 \\ 0 & 0.5 & -3 \\ 0 & 0 & 1 \end{bmatrix}$ b.) $\begin{bmatrix} 0 & -1.5 & 3 \\ 0.25 & 0 & -3 \\ 0 & 0 & 1 \end{bmatrix}$

c.) $\begin{bmatrix} -0.5 & -0.5 & 3 \\ 0.5 & -0.5 & 3 \\ 0 & 0 & 1 \end{bmatrix}$ d.) $\begin{bmatrix} -1 & 0 & 4 \\ 1 & 1 & -4 \\ 0 & 0 & 1 \end{bmatrix}$ e.) $\begin{bmatrix} 2 & 0 & -4 \\ 0 & -2 & 4 \\ 0 & 0 & 1 \end{bmatrix}$

Animation (10 points)

In words, describe how you could use transformation matrices in conjunction with the HTML5 canvas graphics we have studied, to create an animation of a ball rolling across the screen from left to right. Make sure you specify the initial setup, and what happens during the animation. You don't need to give numbers, but you should be as specific as possible about what types of transformations are used, the order they are applied, and any other relevant details.

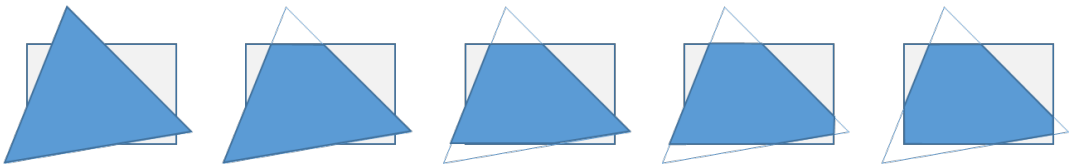
The ball will be rendered using a combination of translation and rotation. We set an initial translation for the starting position of the ball, perhaps offscreen to the left. The rotation matrix can begin as the identity. At each step of the animation we update the translation matrix by applying a small rightward translation, and we update the rotation matrix by applying a small clockwise rotation. We draw the ball by applying the translation first, then the rotation, before drawing. Afterwards we undo both transformations so that the new time we draw the ball we are starting from scratch.

Note that we cannot alternately apply translation and rotation increments because the ball would not then travel in a straight line.

Clipping (20 points)

The drawing canvas is 80 pixels tall by 120 pixels wide. A triangle ABC is to be drawn with its corners at A = (-18,95), B = (32,-30), and C = (132,70).

a.) Assuming that the clipping order is (top, bottom, right, left), draw the polygon after each stage of the Sutherland-Hodgman clipping algorithm.



b.) Compute coordinates for all the corners of the polygon once it is clipped to fit the screen.
Clockwise from bottom left corner: (0,80), (0,50), (20,0), (62,0), (120,58), (120,72), (72,80)

Bézier Splines (16 points)

Consider the spline with control points (in sequential order): (0,1), (4,1), (1,4), (1,0).

a.) Compute the (x,y) coordinates of three points on the curve, at t = 0.2, t = 0.5, and t = 0.8.
(1.64, 1.28), (2, 2), and (1.28, 1.64)
We use the equation $\vec{p} = (1-t)^3\vec{p}_0 + 3t(1-t)^2\vec{p}_1 + 3t^2(1-t)\vec{p}_2 + t^3\vec{p}_3$

b.) Draw a sketch of this curve. *See figure at right.*

c.) The curve section mentioned is parameterized from t = 0 to t = 1. Suppose we want to extrapolate. What are the coordinates of the point at t = -1? At t = 2?
(-43,20) and (20,-43)
This also uses $\vec{p} = (1-t)^3\vec{p}_0 + 3t(1-t)^2\vec{p}_1 + 3t^2(1-t)\vec{p}_2 + t^3\vec{p}_3$

d.) You want to extend the spline by adding another section. The new section should connect smoothly to the first at (1,0) and be a mirror image of it. What are the control points (in sequential order) of the new section?
(1,0), (1,-4), (4,-1), (0,-1)
For smoothness, the new curve needs to start at (1,0) and have its second control point collinear with (1,4). This gives (1,-4). The remaining points are determined by symmetry.