ECE433: COMPUTER GRAPHICS
Fall 2025

Project 1:

Drawing Lines and Curves

(*Deadline: Friday, November 07, 2025, 23:55)*

General Instructions*:*

Use the template to fill your report. You can download it at ....ece433_template.zip....

In the theoretical exercises, write in detail your answers in your own words (do not copy-paste). Do write the sources from which you drew them in the "References" and mention them in the document (as in the template).

In the programming ones, present and explain the algorithm you programmed, and present in detail the experimental results of your programs.
In case you downloaded the code from the web (e.g., GitHub, etc.), point out the source in "References" and explain in detail.
Describe your implementation and the machine you worked on (hardware features, OS, compiler, etc.).
Program in C and Linux, use Makefile to compile / link / execute your programs.
Use the Makefile from the lecture examples and always keep the options for MAC in them.
Report your observations and present the results with tables and graphs.

Graphs and figures must be software-generated and not handmade and scanned.

*In general, your text should be well written and legible, while you should FULLY justify the steps you followed, and comment on the results of each exercise**. (20 credits)***

Submit your work via eclass to a zip file (hw1_id1_id2_id3.zip), which will contain the .pdf file with your solutions, results, and comments (LaTeX generated), and your code (.c, .h files, and your Makefile).

It is noted that the groups remain unchanged throughout the semester.

Exercise 1. (20)

Download the example codes from the lab lectures. Work in a Linux environment. Compile and run all the examples. Change the window size, background color, drawing color, the coordinates of the points, and the projection area. Create screenshots for each case and add them to your report, describing your actions.

Exercise 2. (30)

Extend the Bresenham Line Algorithm so that it can handle general lines of all possible slopes passing through the points $P_0 = (x_0, y_0)$ and $P_1 = (x_1, y_1)$.

1. Algorithm Development

Explain the control (decision) function and identify the decision parameter used in the algorithm.
Describe in detail the derivation process of the extended algorithm.

2. Color Interpolation Extension

Modify the algorithm so that, given the colors at the line endpoints (c_0 for point P_0 and c_1 for point P_1), it computes the appropriate color for each intermediate pixel. Explain in detail.
Assume a smooth color transition based on the RGB color model.

3. Example Test Case

Test your algorithm using the following input: $P_0 = (8, -4)$, c_0 = red = (1, 0, 0), and $P_1 = (-2, 3)$, c_1 = green = (0, 1, 0).
Use the provided pixel grid template (pixels.ppt) to visualize your results.
For each pixel, apply the correct interpolated color using the color selection tools in MS Office / PowerPoint.

4. Report Submission

Include the generated colored line image and the step-by-step calculations of your algorithm in your report.

Clearly present the decision process and color interpolation formula used in each step.

Hint:

Use geometric symmetry (across the axes and/or the main diagonal) to generalize the algorithm for all line slopes efficiently.

Exercise 3. (30)

Design and implement a Bresenham Algorithm for rendering a circle of the form

$$(x - x_0)^2 + (y - y_0)^2 = r^2$$

1. Algorithm Derivation

Develop a Bresenham algorithm suitable for general circles.

Clearly define the decision (control) function and the decision parameter used to determine the pixel positions along the curve.

Provide a detailed description of the derivation process of the algorithm, explaining the mathematical reasoning behind each step.

2. Example Test Case

Test your algorithm with the following parameters (x0, y0) = (3 ,5) , r = 7

Use the provided pixel grid diagram (pixels.ppt) to visualize your results.

For each pixel, apply the color of your choice using the color selection tools in MS Office / PowerPoint.

3. Report Submission

Include in your report:

The final rendered image of the circle.

The detailed step-by-step calculations and decision updates performed by your algorithm.

Clearly present both the mathematical formulation and the practical computational details.

Hint: Use proper symmetry to simplify the computations for the entire circle.

Exercise 4. (50)

Develop a program in C or C++ and OpenGL that implements the algorithm designed in Exercise 2. The program should be capable of drawing lines in all directions with the appropriate color interpolation. During the implementation, you should also consider the aliasing effect (see the attached PDF *"Graphics: Principles & Algorithms"* by Th. Theoharis – A. Boehm) and how it influences the appearance of the drawn lines.

User Interaction Requirements: The user must be able to:

- Select the start and end points of each line using the left mouse button.
- Every time the user selects two points, a new line is drawn.
- Change the color of the next line segment by pressing the keys R/r, G/g, or B/b on the keyboard. The initial drawing color should be red.
- The screen should be cleared only on a right mouse button click.

Program Specifications:

- The window title must be: "Team # – Assignment 1 – Exercise 4" (where # is your team number).
- The window size should be 600×600 pixels.
- The window position on the screen should be at coordinates (# × 10, # × 10), where # is the team number.
- The viewing area should correspond to the square [–300, 300] × [–300, 300].
- The background color should be white.
- Pressing the key Q/q should terminate the program.

In your report for exercise 4, describe:

- The structure of your implementation,
- How the midpoint algorithm was integrated,
- How user interaction and color handling were implemented.
- Add your code to the appendix.

Note: You are not allowed to use OpenGL functions that draw lines (e.g., glBegin(GL_LINES), etc.)

Exercise 5. (50)

Develop a program in C, C++, or OpenGL that implements the algorithm you designed in Exercise 3. The program should be capable of drawing a circle

$$(x - x_0)^2 + (y - y_0)^2 = r^2$$

The user will define the center by clicking in the window and selecting a point. A second click inside the window will define the radius of the circle as the distance between the two selected points.

User Interaction Requirements: The user must be able to:

- Select two points of the interval using the left mouse button.
- Every time the user selects two points, a new circle is drawn.
- The screen should be cleared only on a right mouse button click.
- Change the drawing color using the keys R/r, G/g, and B/b on the keyboard.
- The initial drawing color should be red.

Program Specifications:

- The window title must be: "Team # – Assignment 1 – Exercise 5" (where # is your team number).
- The window size should be 800×800 pixels.
- The window position on the screen should be at coordinates (# × 10, # × 10), where # is the team number.
- The viewing area should correspond to the square [–400, 400] × [–400, 400].
- The background color should be black.
- Pressing the key Q/q should terminate the program.

In your report for exercise 5, describe:

- The structure of your implementation
- The integration of your Bresenham circle algorithm from Exercise 3.
- How user interaction and color handling were implemented.
- Add your code to the appendix.