ECE433: COMPUTER GRAPHICS
Fall 2025

Project 2:
Filling Polygons – Transformations - Clipping
(*Deadline: Sunday, December 7, 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 (hw2_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 (OpenGl_b). 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. (10)
A triangle has vertices $A(0,0), B(2,1), C(1,3)$. Apply the following composite transformation:

1. Rotate the triangle by 90° counterclockwise
2. Scale it by $(1.5, 0.5)$
3. Translate it by $(4, -1)$

Tasks:

- Write the combined matrix $M = T \cdot S \cdot R$
- Compute the final transformed coordinates of points A, B, and C.

Exercise 3. (10)
Rotate the point $P = (3,2)$ by 60° **about the point** $Q = (1,1)$. *(Hint: Positive angles → CCW rotation).*
Tasks:

1. Express the rotation about a point Q as a product of three homogeneous matrices.
2. Compute the resulting coordinates of P.

Exercise 4. (10)
Consider the rectangle with vertices (0,0), (4,0) , (4,2) and (0,2)

1. Apply an **x-shear** with shear factor $k_x = 1.2$.
2. Apply a **y-shear** with shear factor $k_y = -0.5$.

Provide the homogeneous matrices and the final vertex coordinates for both cases.

Exercise 5. (10)
Given the 2D transformation matrix $M = T(2, -3)\ S(2,3)\ R(30°)$ (where multiplication is done in that order),

1. Compute the inverse transformation matrix M^{-1}.
2. Apply M^{-1} to the point $P = (5,5,1)$.

Exercise 6. (10)
A local 2D coordinate system is obtained by:

- rotating the global axes by $30°$, and
- translating the origin to $(2, -1)$.

1. Construct the homogeneous matrix converting **local → global** coordinates.
2. Convert the point $P' = (1,2)$ from local to global coordinates.
3. Convert the point P = (1,1) from global to local coordinates.

Exercise 7. (10)
Given a cube centered at the origin with side length 2, perform the following composite transformation:

1. Scale the cube by (1, 2, 1).
2. Rotate it by $45°$ around the y-axis.
3. Translate it by (3,0,2).

Tasks:

- Write the **combined transformation matrix** $M = T \cdot R_y(45°) \cdot S$
- Compute the transformed coordinates of the cube's vertex $V = (1, 1, 1, 1)$.

Exercise 8. (10)
A local coordinate system has its origin at $O' = (1,2,3)$ and its axes are obtained by rotating the global system by $30°$ around z followed by $45°$ around x.

1. Construct the **homogeneous transformation matrix** converting local coordinates into global coordinates.
2. Convert the local point $P' = (2,0,-1,1)$ into global coordinates.

Exercise 9. (30)

Extend the general polygon-fill algorithm using scanlines (from bottom to top) and different colors on the nodes of the polygons.

1. Algorithm Development

Study the lecture slides and paragraphs 6.10 and 6.11 of the book "Computer Graphics with OpenGL" or 2.6.3 of the book "Graphics and Visualization: Principles and Algorithms" and write a pseudocode. Explain and describe each step of the algorithm.

2. Color Interpolation Extension

Modify the algorithm so that, given the colors at the nodes of the polygon (c_0 for point P_0 and c_1 for point P_1, etc.), it computes the appropriate color for each intermediate pixel of each edge. Then, for each scanline, do interpolation between the ending colors to calculate the colors for each pixel inside the polygon. Explain in detail. Assume a smooth color transition based on the RGB color model.

3. Example Test Case

"Run" and record (as in the lecture slides) step by step your algorithm for the polygon with vertices:

A(3,1), B(1,4), C(5,7), D(9,5), E(7,2) and F(5,3).
If A and C are red, B and E are blue, and D and F have green color in the RGB color model, what is the color of all pixels inside the polygon?
Use the provided auxiliary pixel diagram (file pixels.ppt) to represent the result with the appropriate colors (for each pixel, choose the correct color using the color picker tools in MS Office / PowerPoint).
Hint:
For each scanline, first perform color interpolation along the polygon edges and then, from the two colors (start and end) on the scanline, interpolate to give the appropriate color to the intermediate pixels.

Exercise 10. (30)
Write the Sutherland-Hodgman algorithm (pseudocode) for clipping polygons.
1. Algorithm Development
Study the lecture slides and paragraph 8.8 of the book “Computer Graphics with OpenGL” or 2.9.3 of the book “Graphics and Visualization: Principles and Algorithms” and write a pseudocode. Explain and describe each step of the algorithm.
2. Example Test Case
“Run” and record (as in the lecture slides) step by step your algorithm for the polygon with vertices
A(2,2), B(1,6), C(6,9), D(12,7), E(15,9), F(9,0), G(5,4), and a clipping window defined by bottom-left point (2,3) and top-right point (14,8).
Which are the final vertices of the clipped polygon?
Hint:
Use the provided auxiliary pixel diagram (file pixels.ppt) to represent the results

Exercise 11. (50)
Develop a program in C or C++ and OpenGL that implements the algorithm designed in Exercise 9. The program should be capable of filling polygons with the correct colors.
User Interaction Requirements: The user should be able to:
- Select the nodes of the polygon using the left mouse button. The last node should be selected using the right mouse button. Every time a point is selected, it will also be drawn in your application window
- Change the color of the next point/node by clicking the keys R/r, G/g, B/b, C/c(cyan), Y/y (yellow), and M/m (magenta) 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 2 – Exercise 9" (where # is your team number).
- The window size should be 501×501 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 [0, 500] × [0, 500].
- The background color should be a color of your choice.
- Clicking the key Q/q should terminate the program.

In your report for exercise 4, describe:
- The structure of your implementation,
- How the algorithm was integrated,
- How user interaction and color handling were implemented.
- Add your code to the appendix.
- Create a couple of screenshots of your testing and add them to your report.

<u>Note:</u> You are not allowed to use OpenGL functions that draw polygons (e.g., glBegin(GL_POLYGON), etc.)

Άσκηση 12. (80)
Develop a program in C or C++ and OpenGL that implements the algorithm designed in Exercise 10. Your program should be able to do the clipping of (convex) polygons, while the clipping window is a parallelogram.
User Interaction Requirements: The user must be able to:

- Select the nodes of the polygon using the left mouse button. The last node should be selected using the right mouse button. Every time a point is selected, it will also be drawn in your application window.
- To define a new polygon, the user should clear the application window first (see below).
- Define the clipping window by clicking and dragging the left mouse button. The starting and ending points of this procedure will define the window.
- Alternate between the modes (clipping window definition and polygon definition) by clicking the F1 button.
- The clipping area (even while under selection) will be filled with white color.
- The background color and the color of the polygon to be clipped should be different.
- While defining the clipping window, the whole polygon should be visible.
- Define the clipping window as often as you like. Delete the previously defined clipping window before defining a new one. Therefore, there will be only one active clipping window.

Program Specifications:

- The window title must be: "Team # – Assignment 2 – Exercise 12" (where # is your team number).
- The window size should be 801×601 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 [-4, 4] × [-3, 3].
- The background color should be a grey color of your choice.
- The color of the polygon (to be clipped) should be of your choice and remain the same throughout the execution.
- Clipping should be applied only after clicking the key R/r.
- Clicking the key C/c should clear the application window, so a new example (new polygon, multiple clipping windows) can be solved.
- Pressing the key Q/q should terminate the program.

In your report for exercise 5, describe:

- The structure of your implementation,
- How the algorithm was integrated,
- How user interaction and color handling were implemented.
- Add your code to the appendix.
- Create screenshots of your testing and add them to your report.