

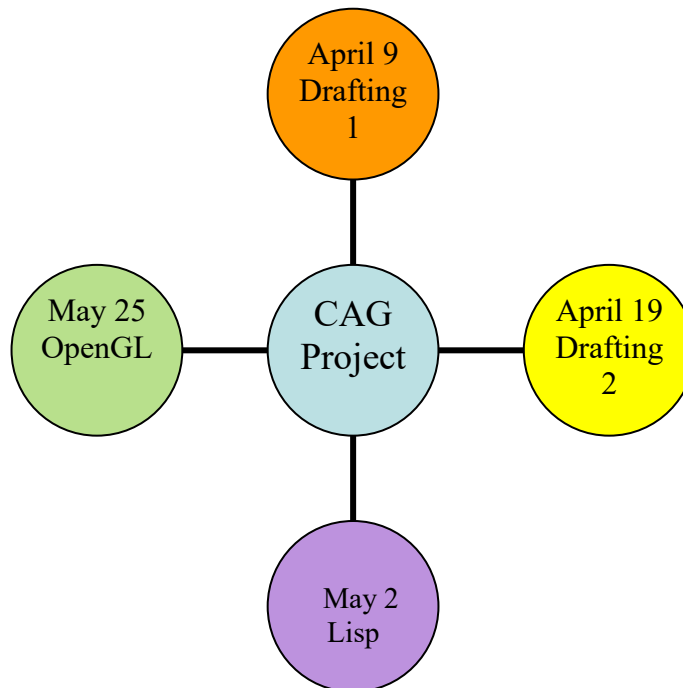
TECHNICAL UNIVERSITY OF CLUJ-NAPOCA

*Computer Aided Graphics - Project
4 parts*

***Technical University of Cluj-Napoca
Faculty of Automation and Computer Science
Department of Automation
Discipline: Computer Aided Graphics***

CAG Project

4 parts - deadlines



Student : Muresan Daria
Group: 30311

Coordinator:
Lecturer Eng. Iulia STEFAN, PhD

Summary

2023-2024

UNIVERSITATEA TEHNICA DIN CLUJ-NAPOCA.....	0
Explanatory Note	Error! Bookmark not defined.
1. Drafting 1	3
1.1 Bibliography Drafting 1	5
2 Drafting 2	6
2.1 Bibliography Drafting 2	9
3 LISP	10
3.1 Bibliography LISP	13
4 OpenGL.....	14
4.1 Bibliography OpenGL	17

1. Drafting 1

1.Object description:

An angle bracket or angle brace or angle cleat is an L-shaped fastener used to join two parts generally at a 90 degree angle. It is typically made of metal but it can also be made of wood or plastic. Angle brackets feature holes in them for screws.

A typical example use of is a shelf bracket for mounting a shelf on a wall. In general, angle brackets have a wide range of applications, and are used, among other things, in building construction, mechanical engineering or to join two pieces of furniture.

2.Commands used

CIRCLE: Draws circles of any size

LINE: Draws straight lines between two points

COPY: Draws a copy of selected objects

MIRROR: Makes mirror images of existing objects

MOVE: Moves designated entities to another location

FILLET: Changes any corner to a rounded corner

TEXT: Draws text characters of any size.

DISTANCE Find the distance between two points in a drawing

LINE Create a simple line

PLINE Make a polyline

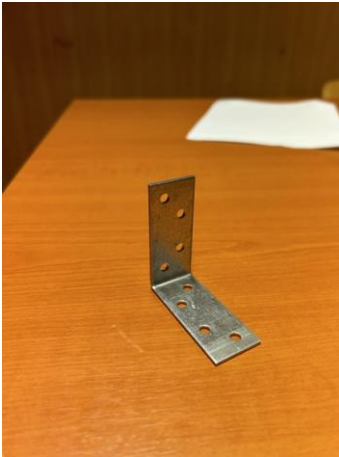
SCALE:Change the scale of an object

DIMLIN:show dimensions on the object

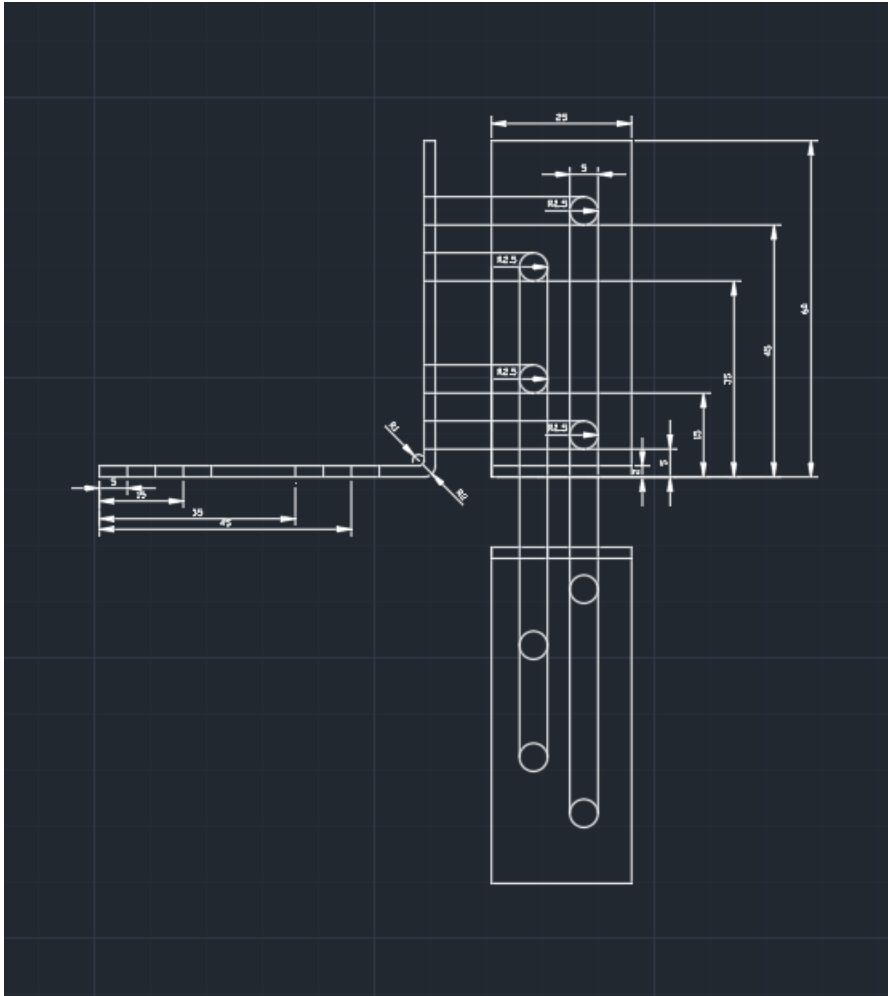
DIMSTYLE: style the text, arrows and dimensions

DIMRAD:show radius of an arc or circle

3.Object photos



4.The DWG file



1.1 Bibliography Drafting 1

- [1] https://classes.engr.oregonstate.edu/cce/winter2020/cce203/CE413_W2020/ACA_D_Basic_Commands.pdf
- [2] <https://www.scan2cad.com/blog/cad/autocad-commands/>
- [3] Laboratory materials
- [4] PDF s from Microsoft Teams

2 Drafting 2

TOOL PALLETES

1) Instrument lines symbols

- Connection to process -blue primary line segment.
- Electric signal line-red dashed line
- Instrument supply-green leader line

2)General instrument or function symbols

Discrete instruments:

- Primary accesible discrete instrument -primary location normally accessible to operator
- Field discrete instrument -field mounted

Shared display,shared control:

- Aux Accessible Discrete Instrument-auxiliary location normally accessible to operator
- FT 65 - FLOW TRANSMITTER: provides electrical outputs that are proportional to flow inputs. They use flow meters to measure the flow of liquids and gases.
- FIC 65- FLOW INDICATOR CONTROLLER: Flow controllers are electric devices which monitor and maintain flow-rate variables, typically in process applications. They can be used in conjunction with pumps and valves in fluid flow systems, in order to provide better control of flow variables.
- PDT 65 – PRESSURE DIFFERENTIAL TRANSMITTER: sensors that use additional electronics to compensate for linearity deviations and temperature errors, outputting measurement results as standardised signals. Every transmitter is measured over the entire pressure and temperature profile and compared to the desired signal span.
- FV 65 - FLOW VALVE: used in pneumatic systems to regulate the flow rate of compressed air. By controlling the flow rate, the speed of the pneumatic cylinder can also be regulated directly. In addition, a good throttling valve contributes to reducing wear due to a lower kinetic load.

- TT 64 - TEMPERATURE TRANSMITTER: converts the signal produced by a temperature sensor into a standard instrumentation signal representing a process variable temperature being measured and controlled.
- TT 66 - TEMPERATURE TRANSMITTER
- TIR 64 + TIR 66-> TEMPERATURE RECORDER INSTRUMENT: At its heart every temperature logger comprises two elements: the temperature sensor or sensors and the recording system that samples the sensor at predetermined intervals and saves the measurement result

3)Valves

- Control valve-directly manipulates the flow of one or more fluid process streams.

4)Equipment and instruments

- tank1-Evaporator: heat exchangers that transfer heat from the process fluid into the refrigerant causing a phase change, evaporation.
- tank2-Knockout drum: a vessel in the flare header designed to remove & accumulate condensed & entrained liquids from the relief gases
- Venturi tube: an engineered primary flow element for the most efficient and accurate differential pressure (DP) flow measurement.
- Centrifugal compressor: elevates gas pressure by adding kinetic energy/velocity to the gas as it flows through an impeller

COMMANDS USED

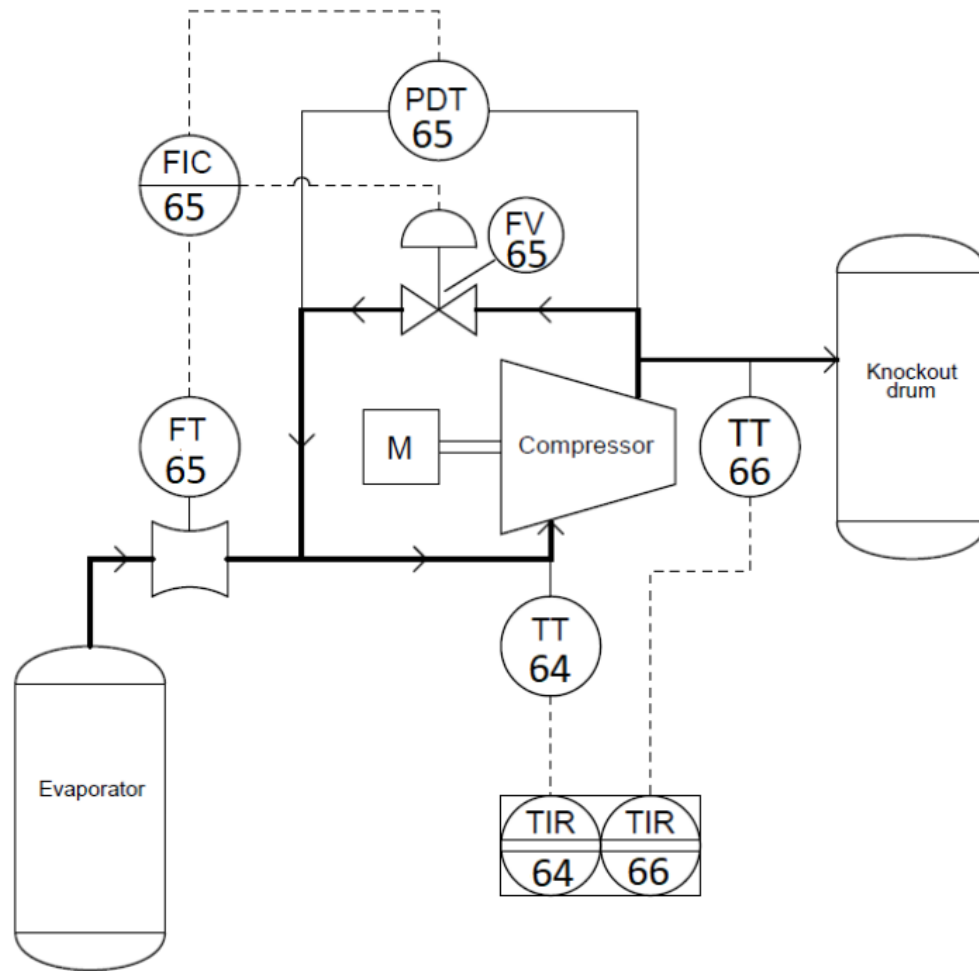
COPY: Draws a copy of selected objects

MOVE: Moves designated entities to another location

TEXT: Draws text characters of any size.

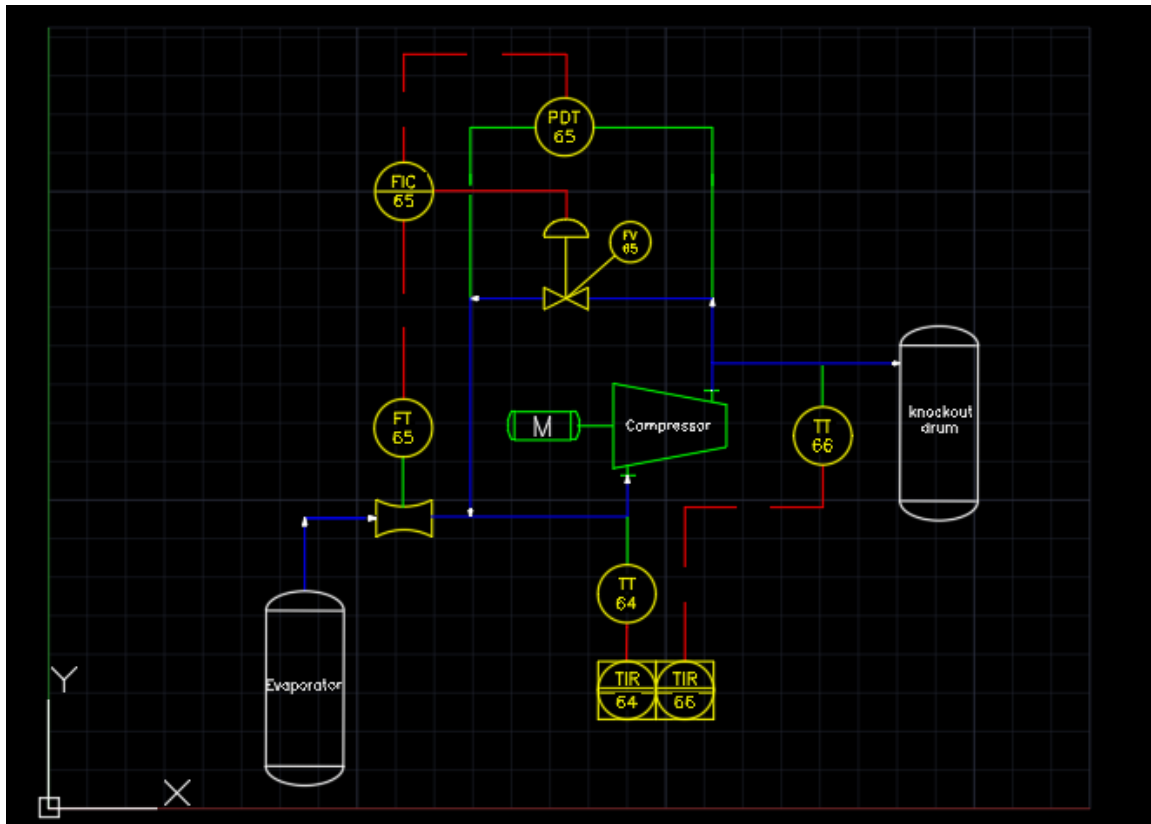
SCALE:Change the scale of an object

Example of process flow diagram:



Here is more instrumentation associated with the compressor than just a flow transmitter. There is also a differential pressure transmitter (PDT), a flow indicating controller (FIC), and a "recycle" control valve allowing some of the vapor coming out of the compressor's discharge line to go back around into the compressor's suction line. Additionally, we have a pair of temperature transmitters reporting suction and discharge line temperatures to an indicating recorder.

Source: <https://www.electricalandcontrol.com/pids-and-loop-diagrams/>



2.1 Bibliography Drafting 2

- [1] Laboratory papers
- [2] <https://www.realpars.com/blog/p-id>
- [3] <https://br.krohne.com/en/products/pressure-measurement/primary-flow-elements/engineered-primary-flow-elements/venturi-tube>
- [4] <https://airoilflaregas.com/pdf/KnockOutDrums.pdf>
- [5] <https://www.thermalcare.com/what-is-an-evaporator-2/>
- [6] <https://control.com/textbook/instrumentation-documents/process-and-instrument-diagrams/>
- [7] <https://symbols.radicasoftware.com/265/pid-iso-equipment>
- [8] https://www.globalspec.com/learnmore/data_acquisition_signal_conditioning/sensor_transmitters/flow_transmitters
- [9] https://keller-druck.com/en/products/pressure-transmitters?gad_source=1&gclid=CjwKCAjwoPOwBhAeEiwAJuXRh8rEakUP5vzxOiErNj58n2-J3G2Izi0UIRN7vdyv-AK4qFKjnjOpBoCjJoQAvD_BwE
- [10] https://www.festo.com/gb/en/e/journal/in-practice/what-is-a-flow-control-valve-id_1516985/
- [11] <https://www.realpars.com/blog/temperature-transmitter>

[12] <https://www.omega.co.uk/technical-learning/introduction-to-temperature-data-logging.html>

3 LISP

FLOWER POWER

Short description of the program.

This AutoCAD Lisp program enables users to create a flower out of already existent petals(blocks). It starts by prompting users to select a color index, determining the color of the petals. Once the color is specified, the program identifies and selects all objects in the drawing that match this color, typically representing the petals of the flower. Users then designate a central point for the flower's arrangement. The program proceeds to relocate each petal to this central point, ensuring a symmetrical layout. Subsequently, it systematically rotates each petal around the central point, creating a radial arrangement akin to a blossoming flower. Throughout the process, the program provides clear instructions and feedback to the user via the command line interface.

The source code with comments.

```
(defun c:floare()
  ;; Prompt the user to enter a color index
  (setq color (getint "\nEnter index for the color you want: 4-blue, 40-orange, 211-pink
  "))

  ;; Check the color index
  (if (and (> color 0) (<= color 255))
    (progn

      ;; selects objects with the specified color index using AutoCAD's ssget function. The
      color index is specified using a DXF group code 62. The selected objects are stored in
      the variable ss as a selection set
      (setq ss (ssget "X" (list (cons 62 color))))

      ;; Check if any objects were selected
      (if ss
        (progn

          ;; specify the center point for the flower
          (setq center (getpoint "\nSelect center for the flower: "))

          ;; Get the number of objects in the selection using the sslength function.
          (setq objCount (sslength ss))
```

```
;; Initialize rotation angle
(setq ang 0)
(setq incrementAngle 75);;there are 5 petals for each color so we divide 360 to 5

;; Loop through each object in the selection set
(setq i 0)
(while (< i objCount)
  (setq obj (ssname ss i)) ; retrieves the name of the object at index i from the
selection set

  ;; Get the object's current base point
  (entget obj): This function is used to retrieve the attributes of the entity referenced
by obj. In the AutoCAD context, an entity can be, for example, a block or a graphical
object.

  (assoc 10 (entget obj)): The assoc function is used to search the list of attributes returned
by entget for the value associated with a specific DXF group code. In this case, we are
looking for the value associated with group code 10, which represents the base
coordinates of the entity (for example, the insertion point for a block or the base point for
a line).

  (cdr (assoc 10 (entget obj))): The cdr function is used to access the data portion of the
key-value pair returned by assoc. In our case, this provides the actual coordinates of the
entity, which are needed to determine its current position in the drawing.      (setq
objCoords (cdr (assoc 10 (entget obj))))

  ;; Move the object to the specified center point
  (command "_move" obj "" "_non" objCoords "_non" center)

  ;; Rotate the object around the center point
  (command "_rotate" obj "" "_non" center ang)

  ;; Increment the rotation angle
  (setq ang (+ ang incrementAngle))

  ;; Increment the index
  (setq i (1+ i))
)

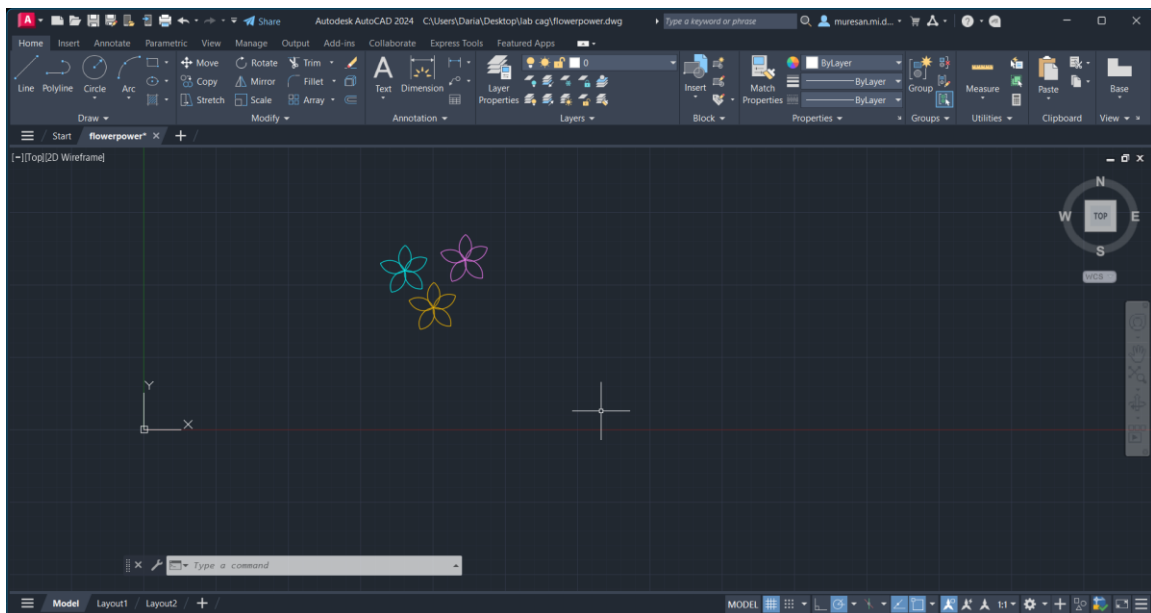
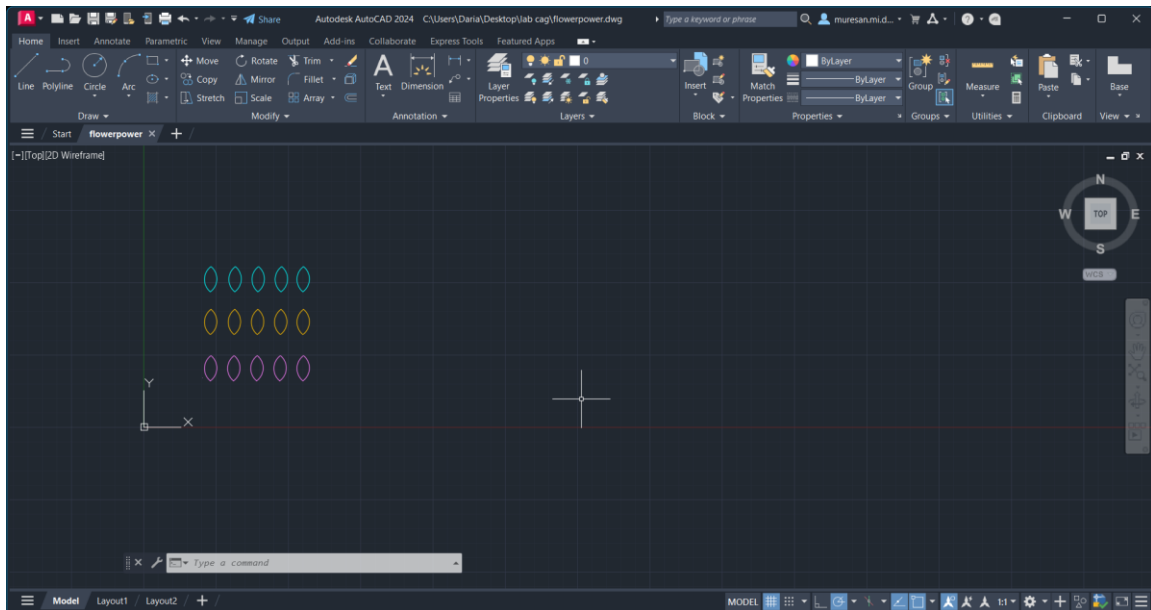
;; Notify the user that the objects have been moved and rotated
(princ " Flower shape <3.")
)
(princ "\nNo objects found with the specified color." ) ; No objects selected
)
```

CAG - Project

```
)  
(princ "\nInvalid color index. Please enter a value between 0 and 255."); Invalid color  
index
```

```
)  
(princ)  
)
```

Print screens with before and after usage of the command



3.1 Bibliography LISP

- [1] <https://blog.projectmaterials.com/flanges/flange-types-piping>
- [2] Laboratory files

4 OpenGL

Objective: The objective of this program is to create an animated simulation of liquid flowing from one tank to another through a connecting pipe.

Description:

This program utilizes OpenGL to create a graphical representation of two tanks and a connecting pipe. The tanks are depicted as rectangular shapes filled with liquid, while the pipe is represented by a line connecting the tanks. The liquid flows from the first tank to the second tank through the pipe. The flow rate of the liquid from the first tank is controlled to create the effect of filling the second tank in synchronization with the flow from the first tank. As the program executes, the liquid level in the first tank progressively rises, depicted by the blue rectangle decreasing in height. Simultaneously, the liquid level in the second tank increases, giving the illusion of the tank filling up.

The source code with comments:

```
#include <windows.h> // Header file for Windows API
#include <GL/gl.h> // Header file for OpenGL
#include <GL/glu.h> // Header file for OpenGL Utility Library
#include <GL/glut.h> // Header file for OpenGL Utility Toolkit
#include <stdlib.h> // Standard library header file for memory allocation,
random numbers, etc.

static GLfloat flow1 = 0.0; // Variable to store the flow level in tank 1
static GLfloat flow2 = 0.0; // Variable to store the flow level in tank 2
static GLfloat flow3 = 0.0; // Variable to store the flow level in tank 3
static GLfloat flow4 = 0.0; // Variable to store the flow level in tank 4

void init(void)
{
    glClearColor(0.0, 0.0, 0.0, 0.0); // Set the background color to black
    glShadeModel(GL_FLAT); // Set the shading model to flat
}

void display(void)
{
    glClear(GL_COLOR_BUFFER_BIT); // Clear the color buffer

    glPushMatrix(); // Push the current matrix stack
```

```

// Tank 1
glColor3f(1.0, 0.40, 0.0); // Set color to orange
glBegin(GL_POLYGON); // Begin drawing a polygon (tank 1)
glVertex2f(-45.0, -40.0); // Define vertices of the tank (bottom-left)
glVertex2f(-25.0, -40.0); // (bottom-right)
glVertex2f(-25.0, 40.0); // (top-right)
glVertex2f(-45.0, 40.0); // (top-left)
glEnd(); // End drawing the polygon

// Liquid in tank 1
glColor3f(0.0, 0.50, 1.0); // Set color to blue
glBegin(GL_POLYGON); // Begin drawing a polygon (liquid in tank 1)
glVertex2f(-45.0, -40.0); // Define vertices of the liquid (bottom-left)
glVertex2f(-25.0, -40.0); // (bottom-right)
glVertex2f(-25.0, 40.0 - flow1); // (top-right) - subtract flow1 to simulate
liquid level
glVertex2f(-45.0, 40.0 - flow1); // (top-left) - subtract flow1 to simulate
liquid level
glEnd(); // End drawing the polygon

// Tank 2, 3, 4 and their respective liquids are drawn similarly (omitted for
brevity)

glPopMatrix(); // Pop the current matrix stack
glutSwapBuffers(); // Swap the front and back buffers to display the
rendered image
}

// The flowDisplay function updates the flow levels of the tanks over time
void flowDisplay(void)
{
    // Update flow levels
    flow1 += 0.005; // Increment flow level for tank 1
    flow2 += 0.02; // Increment flow level for tank 2
    // Similar increments for flow levels of tanks 3 and 4 (omitted for brevity)

    // Check if flow levels exceed maximum capacity, then reset if necessary
    if (flow1 > 75.0)
    {
        flow1 = 75.0;
    }
    // Similar checks for other tanks (omitted for brevity)

    glutPostRedisplay(); // Mark the current window to be redisplayed
}

```

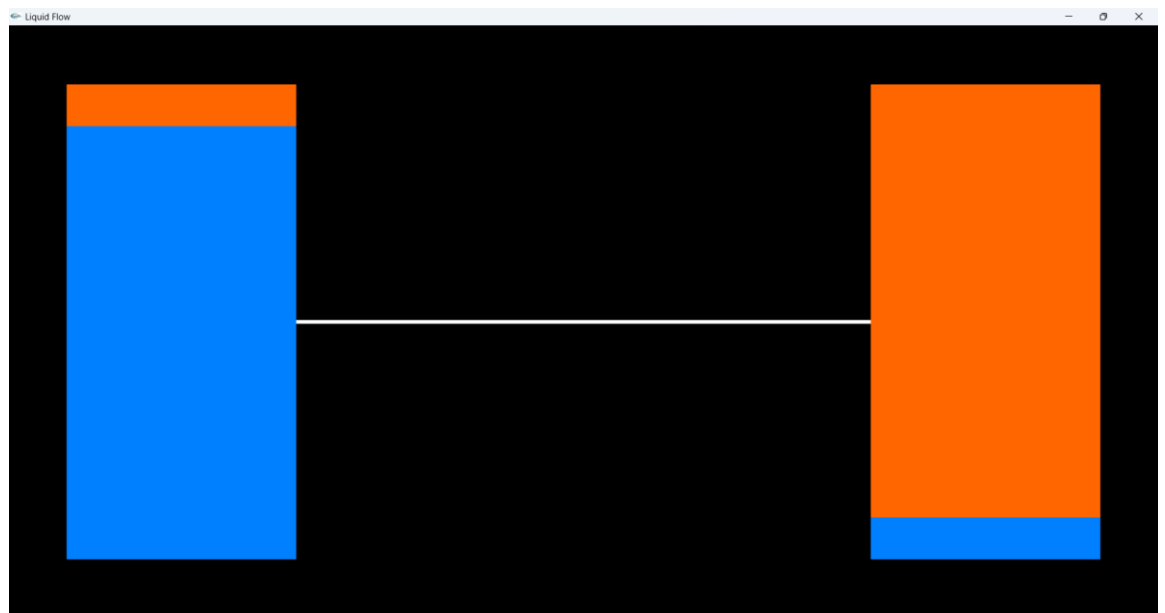
```

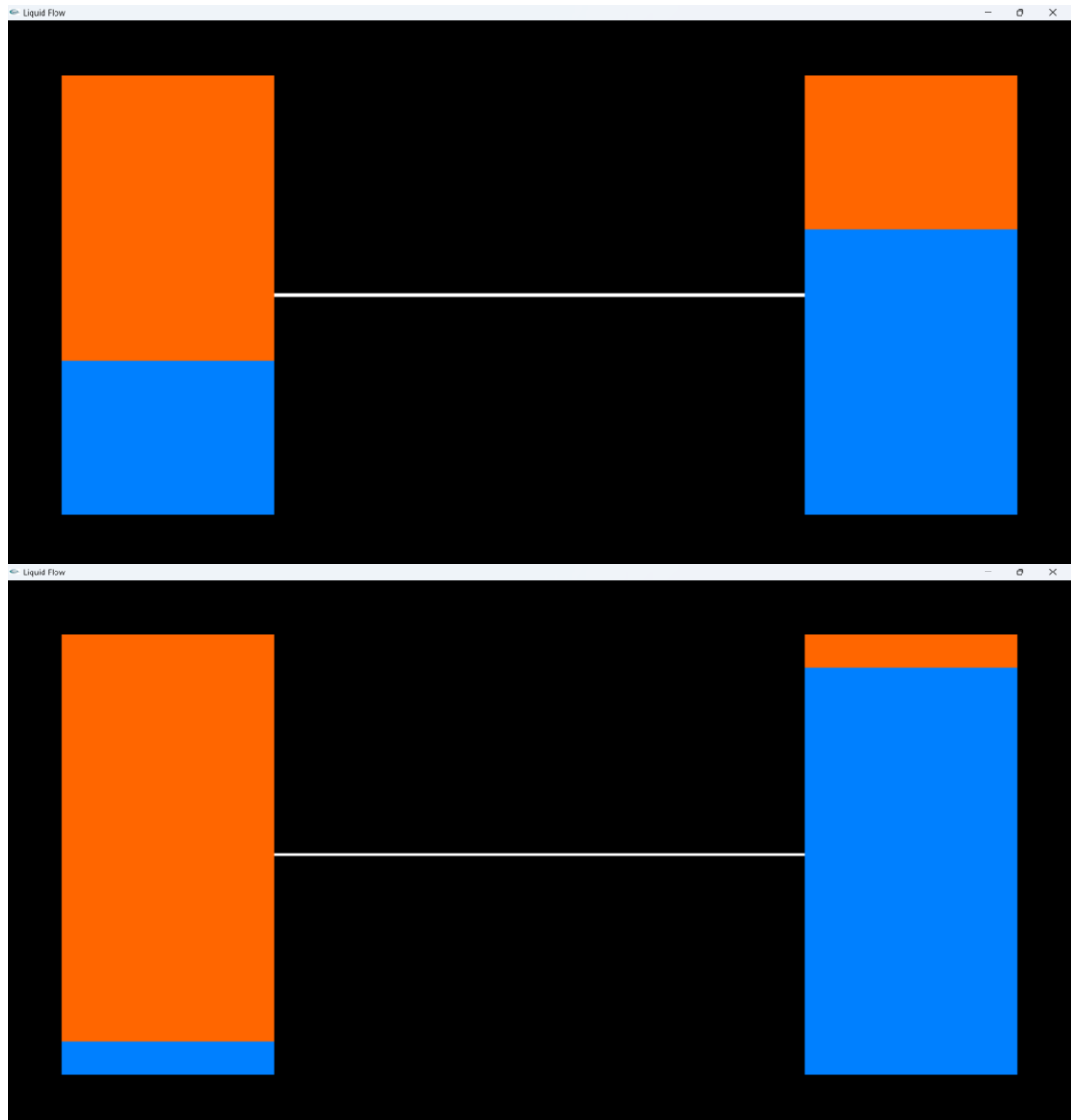
void reshape(int w, int h)
{
    glViewport(0, 0, (GLsizei)w, (GLsizei)h); // Set the viewport to cover the
entire window
    glMatrixMode(GL_PROJECTION); // Set the matrix mode to projection
    glLoadIdentity(); // Load the identity matrix
    glOrtho(-50.0, 50.0, -50.0, 50.0, -1.0, 1.0); // Define a 2D orthographic
projection
    glMatrixMode(GL_MODELVIEW); // Set the matrix mode back to
modelview
    glLoadIdentity(); // Load the identity matrix
}

int main(int argc, char** argv) {
    glutInit(&argc, argv); // Initialize the GLUT library
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB); // Set display mode
(double buffer and RGB color)
    glutInitWindowSize(1500, 600); // Set the initial window size
    glutInitWindowPosition(20, 100); // Set the initial window position
    glutCreateWindow("Liquid Flow"); // Create a window with the given title
    init(); // Call the initialization function
    glutDisplayFunc(display); // Set the display callback function
    glutReshapeFunc(reshape); // Set the reshape callback function
    glutIdleFunc(flowDisplay); // Set the idle callback function
    glutMainLoop(); // Enter the GLUT event processing loop
    return 0; // Return 0 to indicate successful termination
}

```

Print screens of the output window:





4.1 Bibliography OpenGL

- [1] <https://blog.projectmaterials.com/flanges/flange-types-piping>
- [2] Lab materials.