Technical University of Cluj-Napoca

Automation and Computer Science Faculty

Automation and Applied Informatics

Computer-Aided Graphics
CAD In AUTOMATION PROJECT

Student: Coordinating teacher:

Bădău Elena Nicoleta   Goța Dan

GROUP: 30312

2018-2019

Table of contents:

1. First project – Computer-Aided Design of a support

2. Second project – "CONDCIRCLES" An example of function in Autolisp

3. Third project – "P&ID of an automatic thermal system"

4. Fourth project –"OpenGL 3D animation, Rotating Sphere"

1.1 Pictures of the object

<u>Front View</u>

Edge View

1.2 DWG Print Screen

Front View

26,0

52,0

R4,0

A

A

70,0

104,0

Edge View

4,0

44,0

R4,0

Section A - A

8,0

45°

12,0

Surface status	General toleranses		Material	Scale	Projection method
			Steel	2:1	
Designed	Owner		Drawing Name		
Drafted	Bădiţu Elena Nicoleta		Support		
Verified					
Format			Drawing number		
Data					

5

1.3 Table of Commands:

GRID	Modify the grid (on/off and sets a certain precision)
SNAP	Set the snap settings (the precision with which you can navigate in your current project)
UCS	Set the origin and the axes OX and OY (reference system)
PLINE	Creates a 2D polyline
SCALE	Enlarges or reduces selected objects, keeping the proportions
ROTATE	Rotates objects around a basepoint
FILLET	Rounds and fillets the edges of an object
COPY	Works the same way as move, but instead of moving the objects it creates copies of them at the second position
TEXT	Displays text on screen in a certain position
BHATCH	Fills an enclosed area or object with a hatch pattern
BREAK	Breaks the selected object between two pints
CHAMFER	The connection is done by a straight edge intersecting the two objects at a specified distance from their point of intersection
DONUT	Permits the placement of circular rims by introducing the internal diameter, external diameter and the center of the rim
CIRCLE	Allows different ways of creating a circle
DIMANGULAR	Creates an angular dimension
DIMRADIUS	Creates a radius dimention
DIMLINEAR	Creates a linear dimension
DIMSTYLE	Opens a complex dialog box where the style of dimensioning objects may be changed

2.1 The function with a short description


```
Visual LISP for AutoCAD <Drawing1.dwg> - [project.lsp]
 File   Edit   Search   View   Project   Debug   Tools   Window   Help

;;; This function changes the color of circles with radius between 4-5,
;;; hatches the circles with radius between 6-7,
;;; removes the circles with radius between 1-3,
;;;changes the radius of circles which has radius between 8-10.

(defun C:CONDCIRCLES (/ n1 n2  s1 s2 s3 s4  slen1 n b1c c1 entname slen2  b1 c d )
(setq n1 (read "7.0")) ; reads the radius for the radius changing of certain  circles
(setq n2 (read "1"))     ; reads the colorID in order to change the color of certain circles

;; creates the selection sets using filters
(setq s1(ssget "X" '((0 . "circle")(-4 . ">=")(40 . 4.0)(-4 . "<=")(40 . 5.0)))) ; selects all circles with radius between 4-5
(setq s2(ssget "X" '((0 . "circle")(-4 . ">=")(40 . 6.0)(-4 . "<=")(40 . 7.0))))
(setq s3(ssget "X" '((0 . "circle")(-4 . ">=")(40 . 1.0)(-4 . "<=")(40 . 3.0))))
(setq s4(ssget "X" '((0 . "circle")(-4 . ">=")(40 . 8.0)(-4 . "<=")(40 . 10.0))))

;; the color changing of the circles

(setq slen1 (sslength s1))    ; assigns to the variable slen1 the number of entities on the given selection set
  ; iterate over the selection set
(setq n 0)
 (repeat slen1              ; repeats the set of instructions for how many elements are in the selection set
  (setq b1c (entget (ssname s1 n)))   ;retrieves the entity from s1
  (setq n (+ n 1))         ; next
  (setq c1 (cons 62 n2))      ; constructs the new (dotted pair) element
  (setq b1c (append b1c (list c1)))    ; transforms in a list the argument of current entity and the new (dotted pair) element
  (entmod b1c)   ; updates  the entity definition
  )

;; hatching of the circles
(setq n 0)
 (repeat (sslength s2)   ; repeats the set of instructions for how many elements are in the selection set
   (setq entname (ssname s2 n))   ; accesses every entity from a selection set
   (command "-bhatch" "p" "solid" "s" entname "" "") ; hatch command
   (setq n(1+ n))    ; increments n with one
 )
;; removing of the circles
  (setq n 0) ; assigns to the variable n the value 0
 (repeat (sslength s3)
   (entdel (ssname s3 n)) ; deletes an entity from a selection set
   (setq n (1+ n))
 )

;; radius changing of the circles
(setq slen2 (sslength s4)) ; assigns to the variable slen2 the number of entities on the given selection set
(setq n 0)
 (repeat slen2                ;repeats the set of instructions for how many elements are in the selection set
  (setq b1 (entget (ssname s4 n)))   ;retrieves the entity from s4
  (setq n (+ n 1))
  (setq c (assoc 40 b1))       ; searches key 40 in the current entity
  (setq d (cons (car c) n1))    ;constructs the new (dotted pair) element with the first element from the list c and the new radius
  (setq b1 (subst d c b1))    ; modifies the entity through the replacement of the new (dotted pair) element d with c in the current entity
  (entmod b1)   ; updates  the entity definition
 )

  )

Edit: C:/Users/Nico/Desktop/ProiecLisp/project.lsp *  (Visual LISP)
```

The function "CONDCIRCLES" uses four filters to create four different selection sets: the first containing the circles with radius between 4-5, the second

containing the circles with radius between 6-7, the third containing the circles with radius between 1-3 and the fourth containing the circles with radius between 8-10.

There are four main operations, one on every selection set. The user will be able to build as many circles wherever it like with what radius it like and then to call the function "CONDCIRCLES". At the end the circles with radius between >=4 and <=5 change the color in red, the circles with radius between >=6 and <=7 will be hatched, the circles with radius between >=1 and <=3 will be removed and the circles with radius between >=8 and <=10 change the dimension of radius in "7.0". The circles with other radius that are not in the above ranges will remain unchanged.

2.2 The drawing before function call
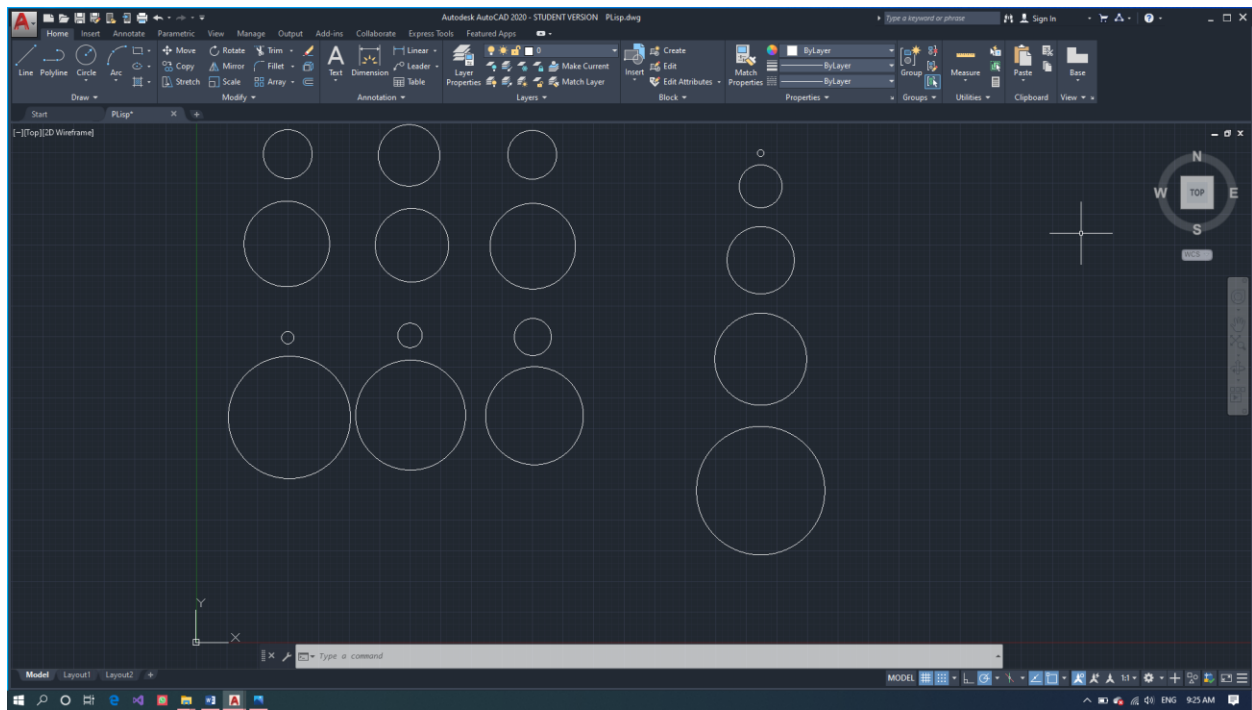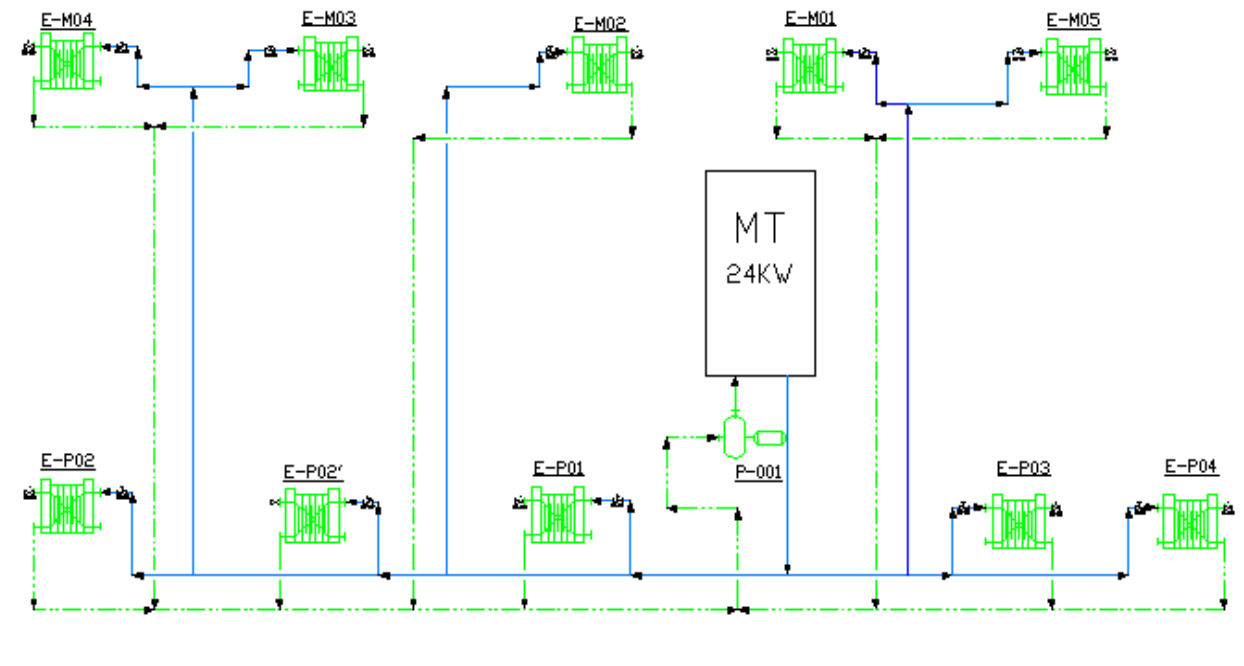
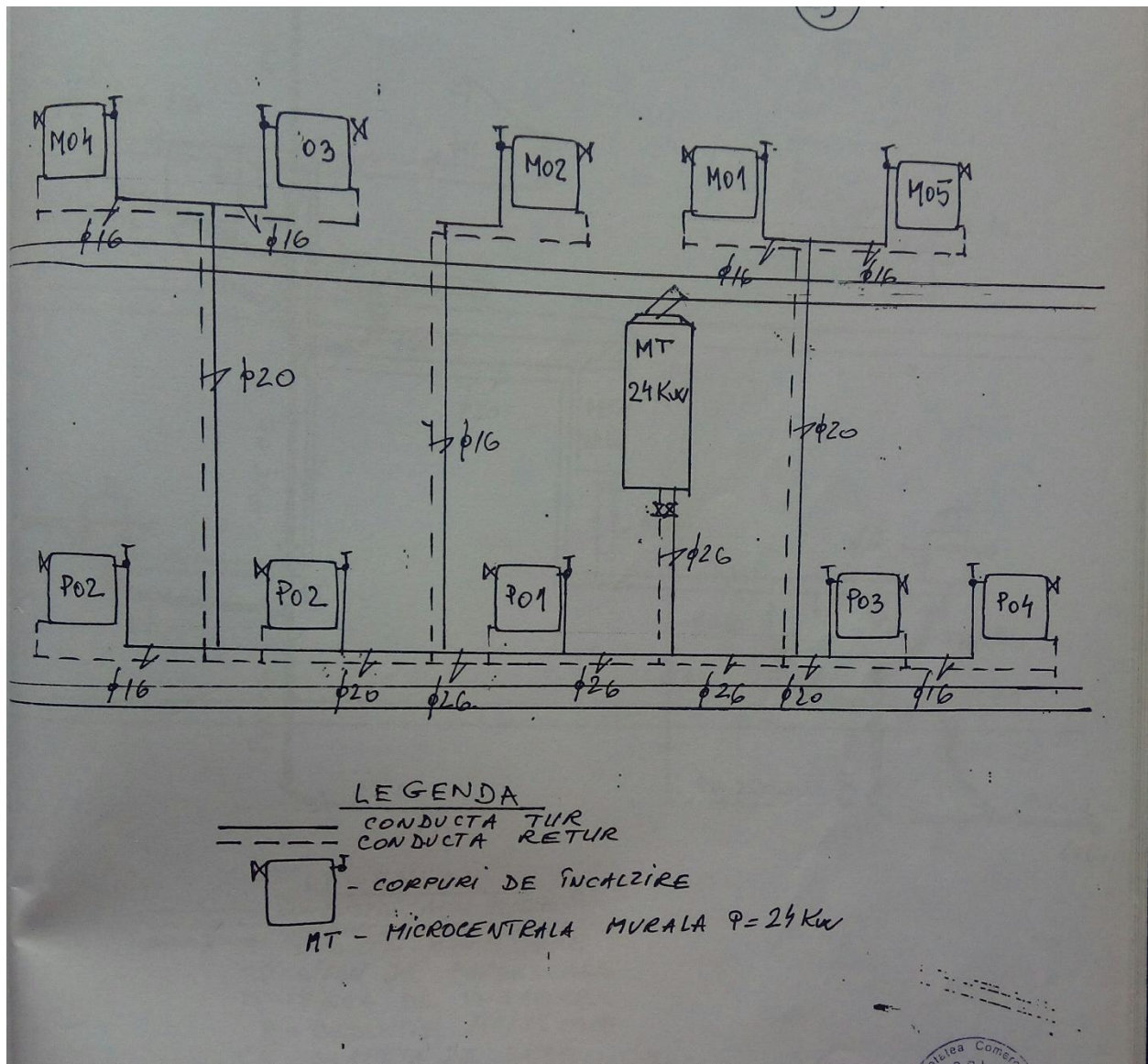## 2.3 The drawing after function call
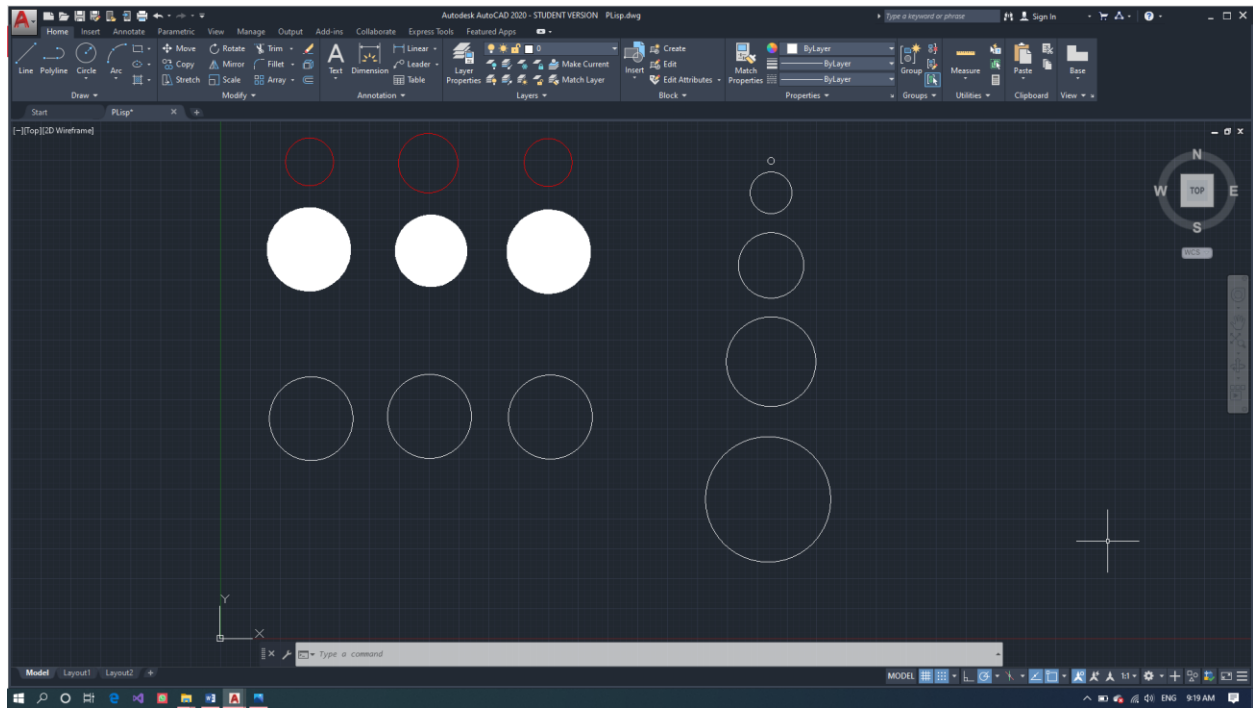
P&ID PROJECT

3.1 Short description

My project illustrates an automatic thermal system of a house, which consists in a "microcentrală murală" and radiators. "Microcentrala murală" has a water tank inside it. The water from the tank is heated with wood resources or gas resources. The thermostat inside it shows when the water inside the tank warmed up, thus it starts the pump which pushes cold water (from radiators) in the water tank. Therefore, heated water comes out through tour pipes and goes to radiators. At the entrance to the radiators exist a stop check valve for each radiator which allows the user to control the heat, if the user doesn't want heat, he can turn off the tap. Also, the radiators have a gate valve for their airing. After that heated water reached in the radiators, it turns back in the water tank through the return pipes, where it is heated again, following the same cicle.

3.2 The drawing

3.3 The image of the scheme

LEGENDA

——————— CONDUCTA TUR
— — — — CONDUCTA RETUR

▢ - CORPURI DE ÎNCALZIRE

MT - MICROCENTRALA MURALA P=24 Kw

OpenGL 3D animation, Rotating Sphere

4.1 Functions roles and a short summary of the program

The project represents a 3D animation of a rotating sphere (spinning sphere) realized in C++, via OpenGL. This project is structured in 6 functions including the main function. These functions can be group in two main categories:

a) Functions which realize the drawing and its animation: displayFunction, idleFunction, reshapeFunction;
b) Functions which realize the human-program interaction: mouseCommand, keyboardCommand.

displayFunction

This function has the role:

- to initialize the matrices used
- to set up the environment
- to assure the rotational, translational movement
- to build the sphere using "glutSolidSphere" predefined command
- to set up the color of the sphere using "glColor3f"

idleFunction

This function has the role to set up the rotation speed and to call the "displayFunction".

reshapeFunction

This is called when the window is reshaped, it scales the object and set up the perspective depending on the window form and dimentions.

mouseCommand

It is responsible for stopping and restarting the rotation of the object and also changing the color of the background when a mouse button is clicked.

keyboardCommand

This function is responsible in order to increase the number of slices and the number of stacks if the user presses "+" button or to decrease the number of slices and the number of stacks if the user presses "-" button until the number 6, also it contains the exit command when "q" is pressed.

Also, this function is responsible for changing the color of the background if the user press one of the keys from 0 to 9 and it contains the exit command when "Esc" is pressed.

main

It fulfills the following roles:

1) Initializes the window, the color of the initial background used and the global variables
2) Calls the previous functions using the specific OpenGL functions and builds the "functionloop" using "glutMainLoop" command. The other OpenGL functions are: glutMouseFunc(), glutKeyboardFunc(), glutIdleFunc(), glutReshapeFunc(), glutDisplayFunc().
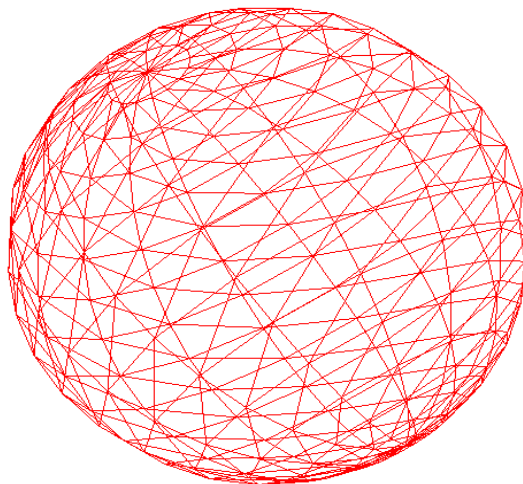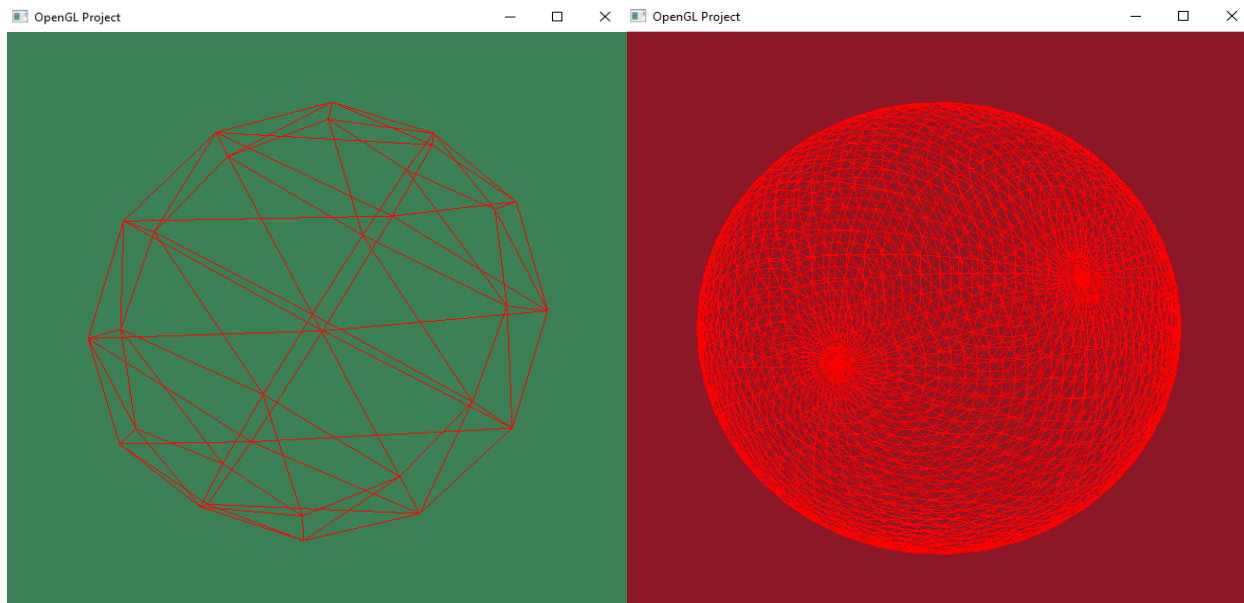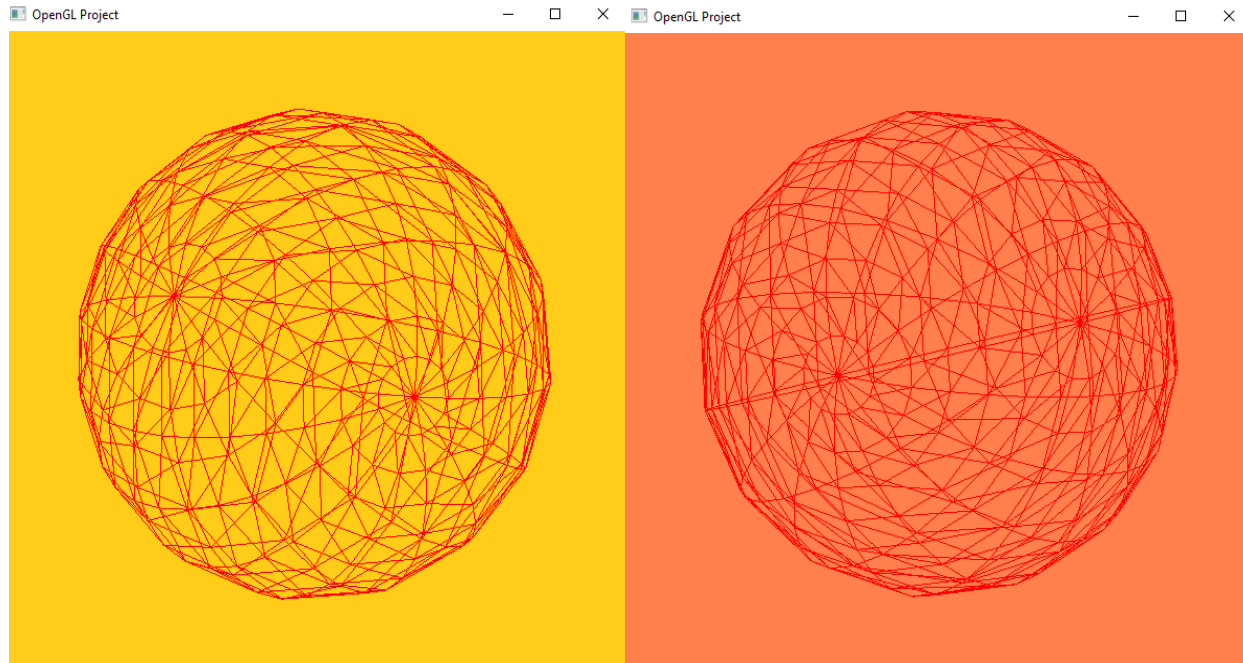

First appearance of the sphere when we run the program
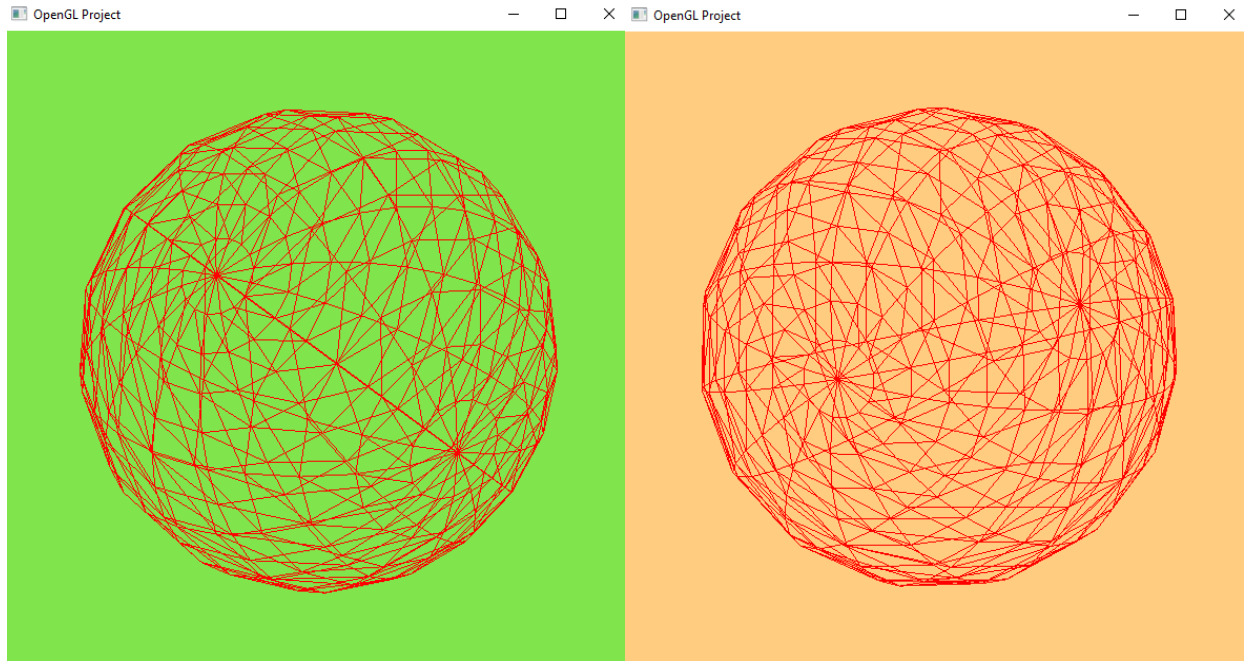
The appearance of the sphere when the user stop the animation using right button of the mouse and restart the animation using the left button of the mouse, changing in the same time the color of the background.

The appearance of the sphere when the user decreases the initial number of the slices and stacks using '-' button and increases the number of the slices and stacks using '+'button, changing in the same time the color of the background using buttons from 0 to 9 for different colors.

The appearance of the sphere when the user wants to change only the color of the background using the buttons from 0 to 9.

4.2 Code

#include <GL\glut.h>

#include <windows.h>

GLfloat xRotated, yRotated, zRotated; /// global variables for the rotation of the sphere

GLdouble radius=1; /// global variable for the initialization of radius

/// static global variables for the initialization of the number of slices and stacks

static int slices = 16;

static int stacks = 16;

/// This is the function displayed by the "glutDisplayFunc()", which draws the body and its movement.

void displayFunction (void)

{

  glMatrixMode(GL_MODELVIEW); /// sets the view point

  glClear(GL_COLOR_BUFFER_BIT); /// clears the drawing buffer and deletes everything on the screen

glLoadIdentity(); /// initializes the matrix before rotating or translating it (before you multiply further matrices with the matrix)

glTranslatef(0.0, 0.0, -5.0); /// translates the draw by z=-5.0, how close or far you see the object

glColor3f(1.0, 0.0, 0.0); /// sets the red color for the segments

/// Changing in the transformation matrix

/// Rotation about:

glRotatef(xRotated, 1.0, 0.0, 0.0); /// x axis

glRotatef(yRotated, 0.0, 1.0, 0.0); /// y axis

glRotatef(zRotated, 0.0, 0.0, 1.0); /// z axis

glutSolidSphere(radius, slices, stacks); /// draws a sphere with 16 slices, 16 stacks and radius=1, built in glut library function

glFlush(); /// makes sure that all the commands are executed and are not preserved in buffer waiting for the OpenGL commands

}

/// This function controls the rotation speed, on the x axis and z axis, incrementing the global variables xRotated and zRotated

void idleFunction (void)

{

  xRotated+=0.01;

  zRotated+=0.01;

  displayFunction(); ///displays again the body and its movement, after the speed is modified

}

void mouseCommand (int button, int state, int x, int y)

{

  /// Using the switch command and the predefined logic variables for mouse, you can stop and restart the animation and also changing the background

  switch(button)

  {

  case GLUT_RIGHT_BUTTON:

    if(state==GLUT_DOWN)

    {

```c
      glutIdleFunc(NULL);

      glClearColor(0.5, 0.9, 0.3, 1.0);

    }

   break;


   case GLUT_LEFT_BUTTON:

     if(state==GLUT_DOWN)

     {

     glutIdleFunc(idleFunction);

     glClearColor(1.0, 0.8, 0.5, 1.0);

     }

   break;


   default:

   break;

  }

}
```

static void keyboardCommand (unsigned char key, int x, int y)

```c
{

   /// Using the switch command, you can increase or decrease the number of slices and stacks

   switch (key)

  {

    case 27 :

    case 'q':

       exit(0);

       break;


    case '+':

       slices++;

       stacks++;

       break;


    case '-':
```

```
        if (slices>6 && stacks>6)

        {

            slices--;

            stacks--;

        }

        break;

    }


    /// Using the switch command and ASCII codes, you can change the background color depending on the key you presses the
buttons from 0 to 9

    switch (key) {

    case 27:

        exit(0);

    case 48:

        glClearColor(0.52, 0.5, 0.26, 1.0);

        break;

    case 49:

        glClearColor(0.23 ,0.5 ,0.34, 1.0);

        break;

    case 50:

        glClearColor(0.7, 0.6, 0.93, 1.0);

        break;

    case 51:

        glClearColor(0.55, 0.8, 0.88, 1.0);

        break;

    case 52:

        glClearColor(0.5, 1.0, 0.5, 1.0);

        break;

    case 53:

        glClearColor(1.0, 1.0, 0.5, 0.0);

        break;

    case 54 :

        glClearColor(1.0, 0.8, 0.1, 1.0);
```

```
          break;

      case 55 :

        glClearColor(1.0, 0.5, 0.68, 1.0);

        break;

      case 56:

        glClearColor(1.0, 0.5, 0.3, 1.0);

        break;

      case 57:

        glClearColor(0.55, 0.09, 0.15, 1.0);

        break;

    }

}
```

/// This function is called when the window is reshaped,it scales the object and set up the perspectives depending on the window form and dimentions

void reshapeFunction(int x, int y)

{

   if(y==0 || x==0) return; /// the window becomes so thin nothing being visible

   glMatrixMode(GL_PROJECTION); /// sets and initializes new projection matrix

   glLoadIdentity();

   /// set up a perspective using: Angle of view: 30 degrees, Near clipping plane distance:0.5, Far clipping plane distance:20.0.

   gluPerspective(30.0, (GLdouble)x/(GLdouble)y, 0.5, 20.0);

   glMatrixMode(GL_MODELVIEW); /// sets the viewpoint

   glViewport(0,0,x,y); /// specifies the viewport rectangle by setting the coordinates of lower left corner and its dimensions

}

int main (int argc, char **argv)

{

   /// Initializing Glut: accesses the specific functions of operating system which are the foundation of any OpenGL program

   glutInit(&argc, argv);

   glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB); /// specifies if the color format is RGBA or another color index, here it sets a unique buffer

   glutInitWindowSize(600, 600); /// sets the window size in pixels

   glutInitWindowPosition(80, 80); /// sets the position of the window

```cpp
    glutCreateWindow("OpenGL Project"); /// creates a new window and sets its title

    glPolygonMode(GL_FRONT_AND_BACK, GL_LINE); /// sets a polygon rasterization mode, draws lines from the front and back
facing polygon's edges

    xRotated=yRotated=zRotated=30.0; /// sets the global variables declared previously

    glClearColor(1.0, 1.0, 1.0, 1.0); /// sets the background color

    glutDisplayFunc(displayFunction); /// register callback handler for window redraw event

    glutReshapeFunc(reshapeFunction); /// register callback handler for window resize event

    glutIdleFunc(idleFunction); /// callback perform background processing tasks or continuous animation when the window
events are not being received


    /// Functions which realize the human-program interaction

    glutMouseFunc(mouseCommand);

    glutKeyboardFunc(keyboardCommand);

    glutMainLoop(); /// enter the infinite event-processing loop

    return 0;

}
```