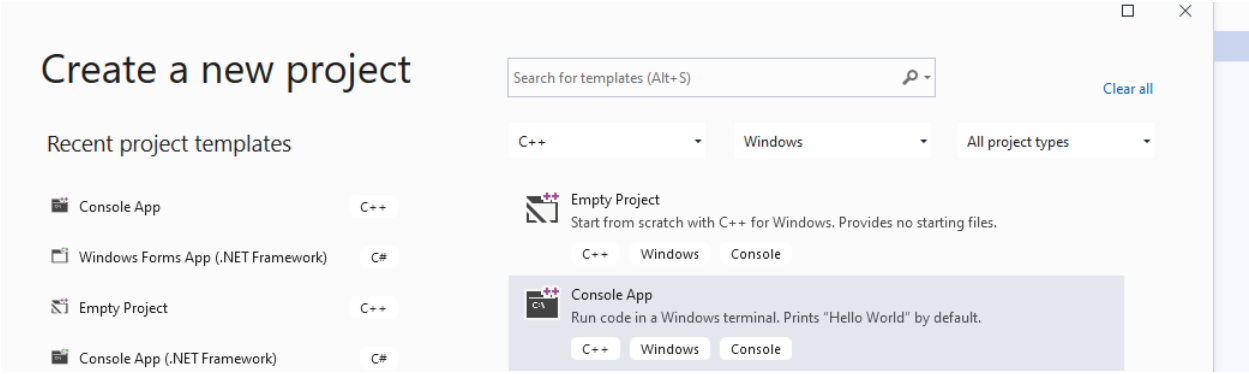
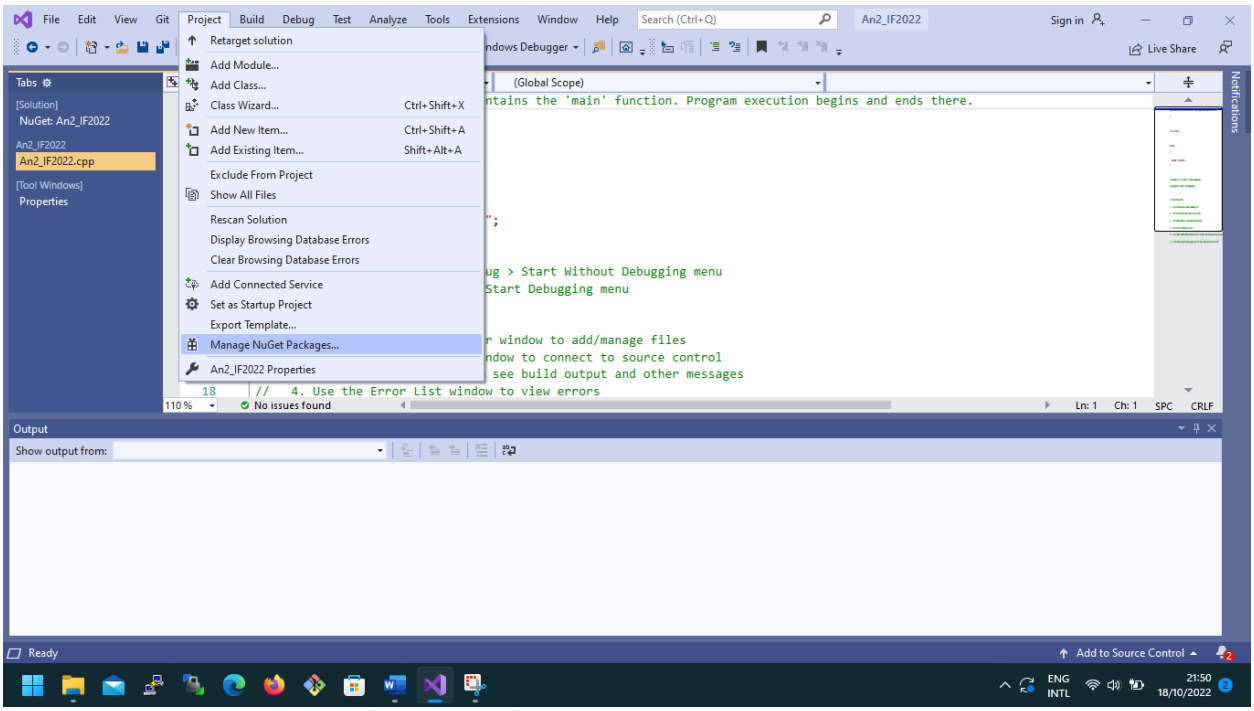
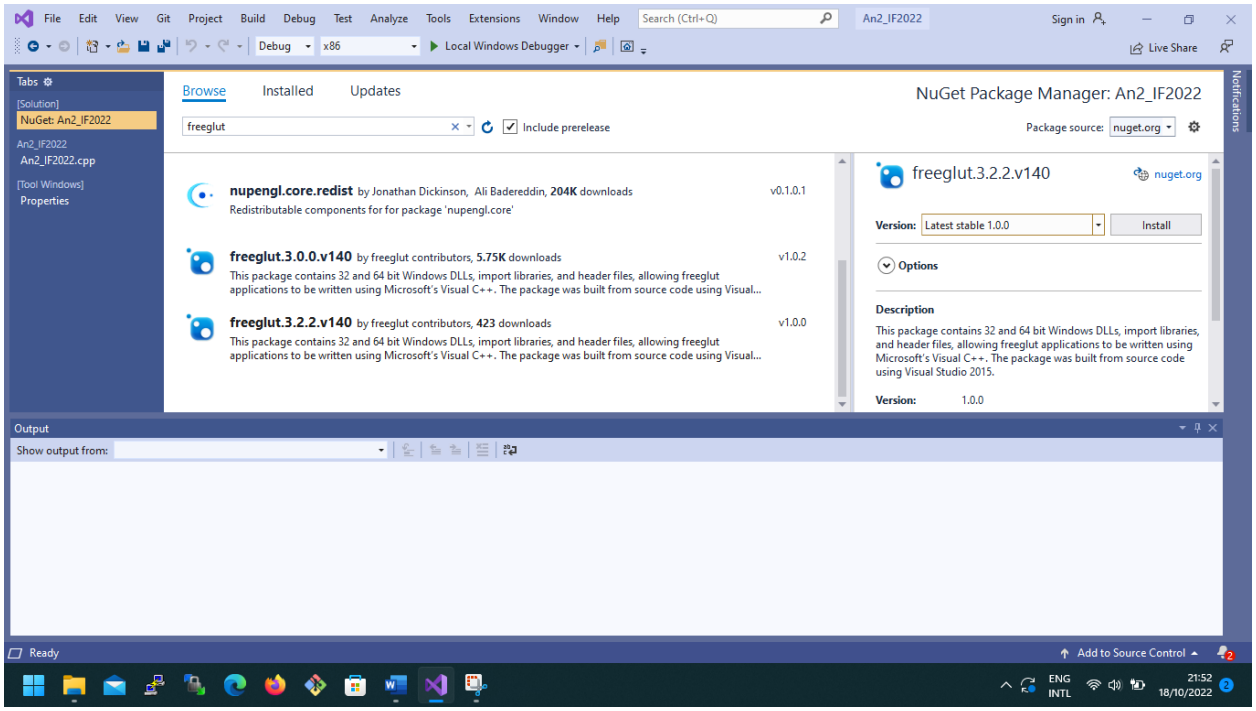
Utilizare freeglut in Microsoft Visual Studio

1.Creare proiect-console application



2. Din Project->Manage NuGet Packages ->Browse ->search freeglut->freeglut-> install si ->nupengl.core->install





Aplicatie Puncte

```
#include <iostream>
#include <gl/freeglut.h>
int dist, i;
//puncte.cpp
void init()
{
	glClearColor(0.0, 0.0, 0.0, 0.0);
	//glPointSize(40.0);
}
void display()
{
	glColor3f(1.0, 1.0, 0.0);
	glPointSize(40.0);
	glBegin(GL_POINTS);

	for (dist = 0,  i = 1; i <= 3; i++)
	{

		glVertex2i(20 * i + dist, 20);
		dist += 40;//y
	}
	glEnd();   glFlush();
}
void reshape(int w, int h)//functia redesenare
{
	glViewport(0, 0, (GLsizei)w, (GLsizei)h);//stabilirea viewportului la dimensiunea
ferestrei
	glMatrixMode(GL_PROJECTION);//specificare matrice modificabila la valoare
argumentului de modificare
	glLoadIdentity();//initializarea sistemului de coordonate
```

```
	gluOrtho2D(0.0, (GLdouble)w, 0.0, (GLdouble)h);//stabileste volumul de vedere folosind o proiectie ortografica
}
void main(int argc, char** argv)
{
	glutInit(&argc, argv);
	glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
	glutInitWindowSize(400, 400);
	glutInitWindowPosition(100, 100);
	glutCreateWindow("puncte");
	init();
	glutDisplayFunc(display);
	glutReshapeFunc(reshape);
	glutMainLoop();

}
```

Aplicatia 2

```
#include <iostream>
#include <gl/freeglut.h>
//puncte.cpp
void init()
{
	glClearColor(0.0, 0.0, 0.0, 0.0);
	//glPointSize(40.0);
}
void display()
{
	glColor3f(1.0, 1.0, 0.0);
	glBegin(GL_POLYGON);//initializare desen poligon
	glVertex2f(0.0, 0.0); //stabilire coordonate triunghi
	glVertex2f(200.0, 200.0);
	glVertex2f(0.0, 200.0);
	glEnd();
	glFlush();
	glPointSize(40.0);
	glColor3f(1.0, 0.0, 0.0);
	glBegin(GL_POINTS);
	glVertex2i(300, 300);
	glVertex2i(20, 20);
	glEnd();   glFlush();
}
void reshape(int w, int h)//functia redesenare
{
	glViewport(0, 0, (GLsizei)w, (GLsizei)h);//stabilirea viewportului la dimensiunea ferestrei
	glMatrixMode(GL_PROJECTION);//specificare matrice modificabila la valoare argumentului de modificare
	glLoadIdentity();//initializarea sistemului de coordonate
	gluOrtho2D(0.0, (GLdouble)w, 0.0, (GLdouble)h);//stabileste volumul de vedere folosind o proiectie ortografica
}
void main(int argc, char** argv)
{
	glutInit(&argc, argv);
	glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
	glutInitWindowSize(400, 400);
```

```
        glutInitWindowPosition(150, 150);
        glutCreateWindow("puncte");
        init();
        glutDisplayFunc(display);
        glutReshapeFunc(reshape);
        glutMainLoop();
    }
```

Aplicatia 3

```
#include <iostream>
#include <gl/freeglut.h>
#include<math.h>
int width = 400;
int height = 400;
int psize = 40;
int distx = 0;
int disty = 0;
void display()
{
        glClear(GL_COLOR_BUFFER_BIT);
        glPointSize(psize);


        glBegin(GL_POINTS);
        disty = 20;
        for (int k = 0; k < 3; k++)
        {
                distx = 20;
                for (int j = 0; j < 3; j++)
                {
                        double r = ((double)rand() / (RAND_MAX));
                        double g = ((double)rand() / (RAND_MAX));
                        double b = ((double)rand() / (RAND_MAX));
                        glColor3d(r, g, b);
                        glVertex2i(40 * j + distx, 40 * k + disty);

                }
                disty += 0;
        }
        glEnd();
        glFlush();

}


void reshape(int w, int h)
{
        glViewport(0, 0, (GLsizei)w, (GLsizei)h); //stabilirea viewportului la dimensiunea
ferestrei
        glMatrixMode(GL_PROJECTION); //specificare matrice modificabila la valoare
argumentului de modificare
        glLoadIdentity(); //initializarea sistemului de coordonate
        gluOrtho2D(0.0, (GLdouble)w, 0.0, (GLdouble)h); //stabileste volumul de vedere
folosind o proiectie ortografica
}//end reshape()

int main(int argc, char** argv)
```

```
{

	glutInit(&argc, argv);
	glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
	glutInitWindowSize(400, 300);
	glutCreateWindow("Puncte");
	glutDisplayFunc(display);
	glutReshapeFunc(reshape);
	glutMainLoop();
	return 0;
}
```

Aplicatia 4

```
#include <iostream>
#include <gl/freeglut.h>

using namespace std;

void Display(void)
{
	glClear(GL_COLOR_BUFFER_BIT);
	glColor3f(0, 1, 0);
	glBegin(GL_LINES);
	// Cadran 1
	for (int i = 0; i < 20; i++)
	{
		glVertex3f(0, 0, 0);
		glVertex3f(1 - i / 20.0, i / 20.0, 0);
	} // Cadran 2
	glColor3f(1, 0.4, 0);
	for (int i = 0; i < 20; i++)
	{
		glVertex3f(0, 0, 0);
		glVertex3f(-1 + i / 20.0, i / 20.0, 0);
	} // Cadran 3
	glColor3f(1, 0.4, 1);
	for (int i = 0; i < 20; i++)
	{
		glVertex3f(0, 0, 0);
		glVertex3f(-1 + i / 20.0, -i / 20.0, 0);
	}
	// Cadran 4
	glColor3f(0.8, 0.4, 0.2);
	for (int i = 0; i < 20; i++)
	{
		glVertex3f(0, 0, 0);
		glVertex3f(1 - i / 20.0, -i / 20.0, 0);
	}
	glEnd();    glFlush();  glColor3f(0, 0, 1);
}
int main(int argc, char** argv) {
	glutInit(&argc, argv);
	glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
```

```
        glutInitWindowSize(600, 600);
        //se specifica modelul de culoare al ferestrei: un singur buffer si culoare RGB
        glutCreateWindow("laborator 2");

        glutDisplayFunc(Display);
        glutMainLoop();
        return 0;
}
```

Aplicatia 5

```
#include <iostream>
#include <gl/freeglut.h>
#include<math.h>
void display()
{
	//glClear(GL_COLOR_BUFFER_BIT);
	glBegin(GL_TRIANGLES);
	glColor3f(1.0, 0.0, 0.0);
	glVertex2i(1, 0);
	glVertex2i(0, 0);
	glVertex2i(0, 1);
	glEnd();
	glPointSize(10.0);
	glColor3f(1, 1, 1);
	glBegin(GL_POINTS);
	for (int i = 0; i < 10; i++) {
		glVertex3f(cos(2 * 3.141 * i / 10.0), sin(2 * 3.14 * i / 10.0), 0);
	}
	glEnd();
	glFlush();
}
int main(int argc, char** argv)
{
	glutInit(&argc, argv);
	glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
	glutInitWindowSize(400, 400);
	glutInitWindowPosition(400, 100);
	glutCreateWindow("aplicatii");
	glutDisplayFunc(display);
	glutMainLoop();
	return 0;
}
```

Aplicatie 4

```
#include <iostream>
#include <gl/freeglut.h>

void init()
{
	glClearColor(0.0, 0.0, 0.0, 0.0);
	glPointSize(40.0);
	glShadeModel(GL_FLAT);
}
void desen()
{
	glClear(GL_COLOR_BUFFER_BIT);
	glColor3f(0.0, 1.0, 0.0);
	glBegin(GL_POLYGON);//initializare desen poligon
	glVertex2f(0.0, 0.0); //stabilire coordonate triunghi
	glVertex2f(200., 0.);
	glVertex2f(200.0, 200.0);//stabilire coordonate triunghi
	glVertex2f(00.0, 200.0);//stabilire coordonate triunghi
	glEnd();//sfisit desenare
			//executare functie
	glFlush();
```

```
	glColor3f(1.0, 1.0, 0.0);
	glBegin(GL_POINTS);
	glVertex2i(100, 300);
	glVertex2i(200, 300);
	glVertex2i(200, 400);
	glColor3f(1.0, 0.0, 1.0);
	glVertex2i(20, 20);
	glEnd();
	glFlush();



}
void reshape(int w, int h)//functia redesenare
{
	glViewport(0, 0, (GLsizei)w, (GLsizei)h);//stabilirea viewportului la dimensiunea
ferestrei
	glMatrixMode(GL_PROJECTION);//specificare matrice modificabila la valoare
argumentului de modificare
	glLoadIdentity();//initializarea sistemului de coordonate
	gluOrtho2D(0.0, (GLdouble)w, 0.0, (GLdouble)h);//stabileste volumul de vedere
folosind o proiectie ortografica
}
void main(int argc, char** argv)
{
	glutInit(&argc, argv);
	glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
	glutInitWindowSize(400, 500);
	glutInitWindowPosition(150, 150);
	glutCreateWindow("Aplicatie Poligon si Puncte");
	init();	glutDisplayFunc(desen);	glutReshapeFunc(reshape);
	glutMainLoop();
	//return 0;
}
```

Aplicatie 6

```
// An2_IF2022.cpp : This file contains the 'main' function. Program execution begins and
ends there.
//

#include <gl/freeglut.h>
void init()//functia initiere
{
	// glClearColor (0.0, 0.0, 0.0, 0.0);//stabileste culoarea de sters
	// glShadeModel (GL_FLAT);
}
void display()//functia de desenare si afisare
{
	glClear(GL_COLOR_BUFFER_BIT);//sterge urmele de desene din fereastra curenta
	glBegin(GL_POLYGON);//initializare desen poligon
	glColor3f(1.0, 0.0, 0.0);//culoarea de desenare
	glVertex2f(200.0, 200.0);//stabilire coordonate triunghi
	glVertex2f(400.0, 200.0);//stabilire coordonate triunghi
	glVertex2f(400.0, 400.0);//stabilire coordonate triunghi
	glEnd();//sfisit desenare
```

```
	glFlush();//executare functie
	glLineWidth(5);
	//glPointSize(40.0);

	glBegin(GL_LINE_LOOP);//initializare desen poligon
	glColor3f(1.0, 1.0, 0.0);//culoarea de desenare
	glVertex2f(200.0, 200.0);//stabilire coordonate triunghi
	glVertex2f(400.0, 0.0);//stabilire coordonate triunghi
	glVertex2f(400.0, 400.0);//stabilire coordonate triunghi
	glEnd();//sfarsit desenare
	glFlush();//executare functie

}
void reshape(int w, int h)//functia redesenare
{
	glViewport(0, 0, (GLsizei)w, (GLsizei)h);//stabilirea viewportului la dimensiunea
ferestrei
	glMatrixMode(GL_PROJECTION);//specificare matrice modificabila la valoare
argumentului de modificare
	glLoadIdentity();//initializarea sistemului de coordonate
	gluOrtho2D(0.0, (GLdouble)w, 0.0, (GLdouble)h);//stabileste volumul de vedere
folosind o proiectie ortografica
}
int main(int argc, char** argv) //creare fereastra
{
	glutInit(&argc, argv);
	glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);//se specifica modelul de culoare al
ferestrei: un singur buffer si culoare RGB
	glutInitWindowSize(600, 600);//initiaza dimensiunea ferestrei principale 600x600
pixeli
	glutInitWindowPosition(200, 10);//initiaza in fereastra principala fereastra de
afisare
	glutCreateWindow("TRIUNGHIURI");
	init();
	glutDisplayFunc(display);//se reimprospateza continutul ferestrei
	glutReshapeFunc(reshape);//functia redesenare
	glutMainLoop();//bucla de procesare a evenimentelor
	return 0;
}
```

Aplicatie cu functie de rotatie

```
#include <GL/glut.h>
#include <stdlib.h>

#include <iostream>
#include <gl/freeglut.h>
void roteste_Y(int p_grade)
{
	glRotatef(p_grade, 0.0, 1.0, .0);
	glutPostRedisplay();
}
void roteste_X(int p_grade)
{
	glRotatef(p_grade, 0., 1., .0);
	glutPostRedisplay();
}
void OnKeyPress(unsigned char key, int x, int y)
{
	if (key == 27)
		exit(0);
	switch (key)
	{
	case 'q':
	case 'Q':

		roteste_Y(3);
		break;
	case 'w':
	case 'W':
		roteste_Y(-3);
		break;

	case 'a':
	case 'A':

		roteste_X(3);
		break;
	case 's':
	case 'S':
		roteste_X(-3);
		break;
	}
}
void OnMouseClick(int button, int state, int x, int y)
{
	if (button == GLUT_LEFT_BUTTON && state == GLUT_DOWN)
	{
		roteste_Y(20);
	}
	if (button == GLUT_RIGHT_BUTTON && state == GLUT_DOWN)
	{
		roteste_Y(-20);
	}
}

void display(void)
```

```
{

	glClear(GL_COLOR_BUFFER_BIT);

	int l = 10;

	for (double i = 0; i <= l; i++) {
		glBegin(GL_LINE_LOOP);
		glColor3f(1 - i / 10, i / 20, 1);
		glVertex3f(1 - i / l, 0, 0);
		glVertex3f(0, 1 - i / l, 0);
		glVertex3f(-(1 - i / l), 0, 0);
		glVertex3f(0, -(1 - i / l), 0);
		glEnd();
	}

	glFlush();

}

int main(int argc, char** argv) {
	glutInit(&argc, argv);
	glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);//se specifica modelul de culoare al
ferestrei: un singur buffer si culoare RGB
	glutCreateWindow("Curs 19.10.2022");
	glutKeyboardFunc(OnKeyPress);
	glutMouseFunc(OnMouseClick);

	glutDisplayFunc(display);
	glutMainLoop();
	return 0;
}
```