МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ

НАФІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

ФАКУЛЬТЕТ ІНФОРМАТИКИ І ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

КАФЕДРА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

Лабораторна робота №3

з дисципліни «Комп’ютерна графіка»

Виконав:

студент гр. ІО-41

бригада №5

Смішний Дмитро

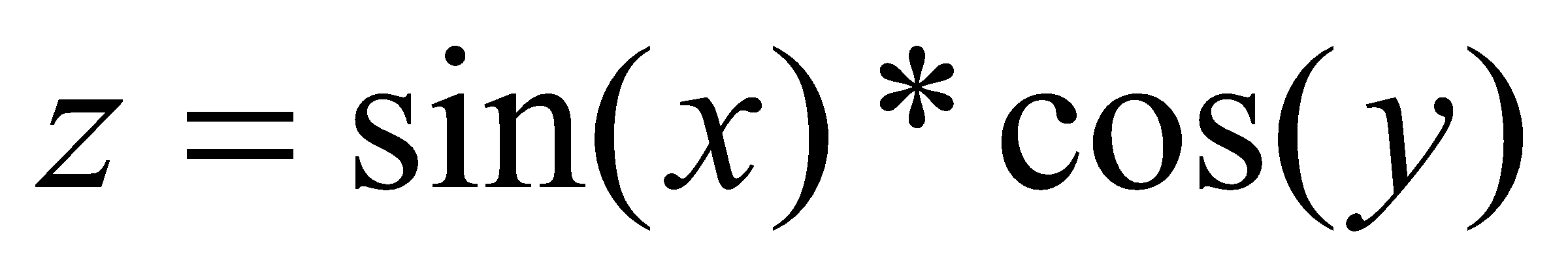
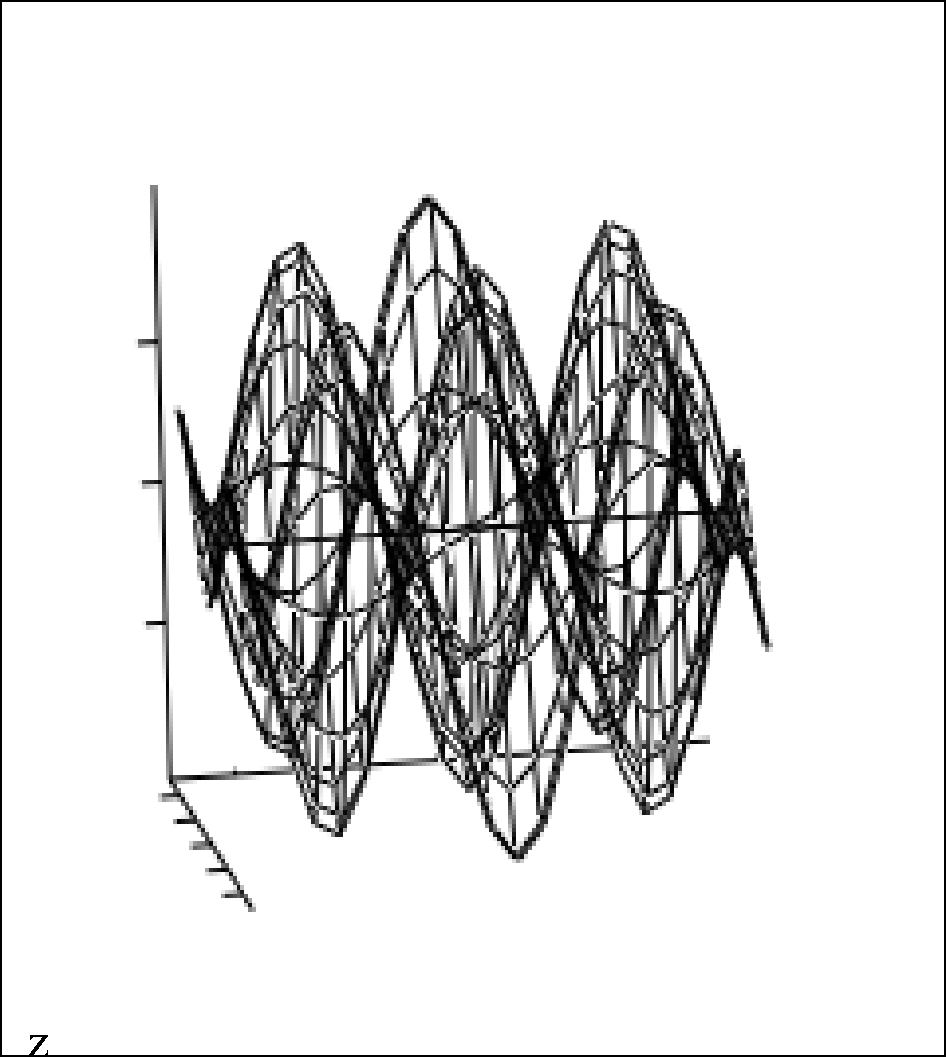
Перевірив:

Саверченко В. Г.

Київ 2016

**Варіант завдання**: 5

Відповідно до варіанта завдання потрібно програмними засобами побудувати поверхню, задану функцією.

**Код програми**:

**Клас MainWindow**

namespace CompGraphLab3

{

/// <summary>

/// Interaction logic for Window1.xaml

/// </summary>

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

}

// The main object model group.

private Model3DGroup MainModel3Dgroup = new Model3DGroup();

// The camera.

private PerspectiveCamera TheCamera;

// The camera's current location.

private double CameraPhi = Math.PI / 6.0; // 30 degrees

private double CameraTheta = Math.PI / 6.0; // 30 degrees

private double CameraR = 22.0;

// The change in CameraPhi when you press the up and down arrows.

private const double CameraDPhi = 0.1;

// The change in CameraTheta when you press the left and right arrows.

private const double CameraDTheta = 0.1;

// The change in CameraR when you press + or -.

private const double CameraDR = 1;

// Create the scene.

// MainViewport is the Viewport3D defined

// in the XAML code that displays everything.

private void Window\_Loaded(object sender, RoutedEventArgs e)

{

// Give the camera its initial position.

TheCamera = new PerspectiveCamera();

TheCamera.FieldOfView = 60;

MainViewport.Camera = TheCamera;

PositionCamera();

// Define lights.

DefineLights();

// Create the model.

DefineModel(MainModel3Dgroup);

// Add the group of models to a ModelVisual3D.

ModelVisual3D model\_visual = new ModelVisual3D();

model\_visual.Content = MainModel3Dgroup;

// Add the main visual to the viewportt.

MainViewport.Children.Add(model\_visual);

}

// Define the lights.

private void DefineLights()

{

AmbientLight ambient\_light = new AmbientLight(Colors.Gray);

DirectionalLight directional\_light =

new DirectionalLight(Colors.Gray, new Vector3D(-1.0, -3.0, -2.0));

MainModel3Dgroup.Children.Add(ambient\_light);

MainModel3Dgroup.Children.Add(directional\_light);

}

// Add the model to the Model3DGroup.

private void DefineModel(Model3DGroup model\_group)

{

// Make a mesh to hold the surface.

MeshGeometry3D mesh = new MeshGeometry3D();

// Make the surface's points and triangles.

const double xmin = -8;

const double xmax = 8;

const double dx = 0.05;

const double ymin = -8;

const double ymax = 8;

const double dy = 0.05;

for (double x = xmin; x <= xmax - dx; x += dx)

{

for (double y = ymin; y <= ymax - dy; y += dx)

{

// Make points at the corners of the surface

// over (x, z) - (x + dx, z + dz).

Point3D p00 = new Point3D(x, F(x, y), y);

Point3D p10 = new Point3D(x + dx, F(x + dx, y), y);

Point3D p01 = new Point3D(x, F(x, y + dy), y + dy);

Point3D p11 = new Point3D(x + dx, F(x + dx, y + dy), y + dy);

// Add the triangles.

AddTriangle(mesh, p00, p01, p11);

AddTriangle(mesh, p00, p11, p10);

}

}

// Make the surface's material using a solid orange brush.

DiffuseMaterial surface\_material = new DiffuseMaterial(Brushes.Purple);

// Make the mesh's model.

GeometryModel3D surface\_model = new GeometryModel3D(mesh, surface\_material);

// Make the surface visible from both sides.

surface\_model.BackMaterial = surface\_material;

// Add the model to the model groups.

model\_group.Children.Add(surface\_model);

}

// The function that defines the surface we are drawing.

private double F(double x, double z)

{

return Math.Cos(x)\*Math.Sin(z);

}

// Add a triangle to the indicated mesh.

private void AddTriangle(MeshGeometry3D mesh, Point3D point1, Point3D point2, Point3D point3)

{

// Get the points' indices.

int index1 = AddPoint(mesh.Positions, point1);

int index2 = AddPoint(mesh.Positions, point2);

int index3 = AddPoint(mesh.Positions, point3);

// Create the triangle.

mesh.TriangleIndices.Add(index1);

mesh.TriangleIndices.Add(index2);

mesh.TriangleIndices.Add(index3);

}

// Create the point and return its new index.

private int AddPoint(Point3DCollection points, Point3D point)

{

// Create the point and return its index.

points.Add(point);

return points.Count - 1;

}

// Adjust the camera's position.

private void Window\_KeyDown(object sender, KeyEventArgs e)

{

switch (e.Key)

{

case Key.Up:

CameraPhi += CameraDPhi;

if (CameraPhi > Math.PI / 2.0) CameraPhi = Math.PI / 2.0;

break;

case Key.Down:

CameraPhi -= CameraDPhi;

if (CameraPhi < -Math.PI / 2.0) CameraPhi = -Math.PI / 2.0;

break;

case Key.Left:

CameraTheta += CameraDTheta;

break;

case Key.Right:

CameraTheta -= CameraDTheta;

break;

case Key.Add:

case Key.OemPlus:

CameraR -= CameraDR;

if (CameraR < CameraDR) CameraR = CameraDR;

break;

case Key.Subtract:

case Key.OemMinus:

CameraR += CameraDR;

break;

}

// Update the camera's position.

PositionCamera();

}

// Position the camera.

private void PositionCamera()

{

// Calculate the camera's position in Cartesian coordinates.

double y = CameraR \* Math.Sin(CameraPhi);

double hyp = CameraR \* Math.Cos(CameraPhi);

double x = hyp \* Math.Cos(CameraTheta);

double z = hyp \* Math.Sin(CameraTheta);

TheCamera.Position = new Point3D(x, y, z);

// Look toward the origin.

TheCamera.LookDirection = new Vector3D(-x, -y, -z);

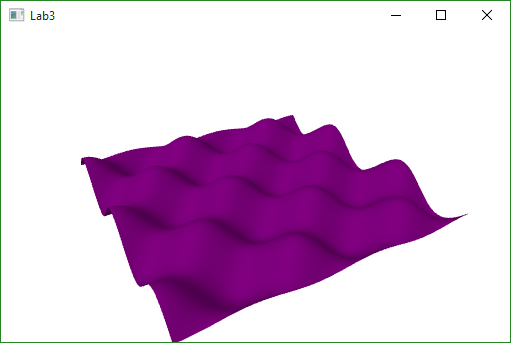
// Set the Up direction.

TheCamera.UpDirection = new Vector3D(0, 1, 0);

}

}

}

**Результати роботи програми:**

**Висновок:** На основі раніше викладеного матеріалу з дисципліни «Комп’ютерна графіка» і власних знань, здобутих в процесі вивчення мови програмування C#, було побудовано поверхню, задану відповідно до варіанту, використовуючи стандартні засоби мови. були закріплені теоретичні знання, отримані на лекціях з даного предмету